

OTOMASI DASAR DENGAN PYTHON

INTERMEDIATE CLASS | SERI I4 | KEHUTANAN & PERTANIAN



DAFTAR ISI

Pendahuluan: Dari Klik ke Skrip: Menyuruh QGIS Bekerja Sendiri	1
Untuk siapa buku ini	1
Cara membaca buku ini	1
Kenalan lagi dengan KPH Contoh	1
Perangkat dan data	2
Menyiapkan folder	2
Hubungan dengan seri lain	2
Hak cipta dan penggunaan	3
Unduhan: buku dan paket data	3
BAB 1: Python di Dalam QGIS: Bicara dengan Peta Lewat Perintah	4
Studi kasus: "Daftar petak sebelum rapat"	4
Konsep: Python di QGIS dalam tiga kalimat	4
Bagian A: PyQGIS di QGIS	5
A.1 Membuka Konsol Python	5
A.2 Lima hal Python yang cukup untuk memulai	5
A.3 Memuat layer dan mengambilnya dengan nama	6
A.4 Membaca fitur dan atribut	8
A.5 Menyimpan proyek dan membukanya lagi	9
Bagian B: ArcPy dasar di ArcGIS Pro	10
Bagian C: ArcPy dasar di ArcMap 10.8	11
BAB 2: Alat Processing dari Skrip: Satu Perintah, Satu Analisis	14
Studi kasus: "Sempadan sungai tiap petak"	14
Konsep: alat Processing dalam tiga kalimat	14
Bagian A: processing.run di QGIS	15
A.1 Menemukan ID alat dan parameternya	15
A.2 Keluaran sementara: layer atau teks?	16
A.3 Rantai alat vektor	17
A.4 Alat raster: NDVI, tinggi tajuk, dan statistik zonal	18
Bagian B: Alat geoprocessing lewat ArcPy di ArcGIS Pro	19
Bagian C: Alat geoprocessing lewat ArcPy di ArcMap 10.8	20
BAB 3: Skrip Batch: Satu Resep untuk Banyak Berkas	23
Studi kasus: "Lima puluh ubin foto drone"	23
Konsep: batch dalam tiga kalimat	23
Bagian A: Batch di QGIS	24
A.1 Tanpa menulis kode: batch bawaan	24
A.2 Skrip batch sederhana	24
A.3 Satu putaran untuk tiap fitur	25
A.4 Batch di latar belakang dengan QgsTask	26
Bagian B: Batch di ArcGIS Pro	28
Bagian C: Batch di ArcMap 10.8	29
BAB 4: Penanganan Galat: Skrip yang Tidak Mudah Menyerah	31
Studi kasus: "Kiriman dari enam tim"	31
Konsep: penanganan galat dalam tiga kalimat	31

Bagian A: Penanganan galat di QGIS.....	32
A.1 Memeriksa sebelum memakai.....	32
A.2 Batch tahan galat dengan laporan.....	33
A.3 Menangkap galat dari alat Processing.....	35
Bagian B: Penanganan galat di ArcGIS Pro.....	36
Bagian C: Penanganan galat di ArcMap 10.8.....	37
BAB 5: Ekspresi: Bahasa Rumus yang Dipakai di Mana-Mana.....	40
Studi kasus: "Kalimat laporan yang menulis dirinya sendiri".....	40
Konsep: ekspresi dalam tiga kalimat.....	40
Bagian A: Ekspresi di QGIS.....	41
A.1 Ekspresi dasar.....	41
A.2 Menghitung ekspresi dari Python.....	41
A.3 Agregasi: menghitung dari layer lain.....	43
A.4 Ekspresi di dalam alat Processing.....	43
Bagian B: Ekspresi di ArcGIS Pro.....	45
Bagian C: Ekspresi di ArcMap 10.8.....	45
BAB 6: Modeler Lanjut: Mesin dengan Tombol Pengaturan.....	47
Studi kasus: "Sempadan 20, 30, atau 50 meter?".....	47
Konsep: model lanjut dalam tiga kalimat.....	47
Bagian A: Model Designer di QGIS.....	48
A.1 Menyusun model dengan klik.....	48
A.2 Isian model dan ekspresi.....	48
A.3 Penjaga: menghentikan model dengan pesan jelas.....	49
A.4 Membangun model lewat Python.....	49
A.5 Menjalankan model dari Python.....	51
A.6 Dari model ke skrip.....	52
Bagian B: ModelBuilder di ArcGIS Pro.....	52
Bagian C: ModelBuilder di ArcMap 10.8.....	52
BAB 7: Bab Jembatan ke Master: Proyek Mini Terpadu dan Ceklis Masuk M1.....	54
Studi kasus: "Laporan awal bulan, satu skrip".....	54
Konsep: jembatan dalam tiga kalimat.....	54
Bagian A: Jembatan di QGIS.....	55
A.1 Rancangan proyek.....	55
A.2 Skrip terpadu.....	55
A.3 Versi latar belakang.....	58
A.4 Peta celah: apa yang M1 pakai, dan di mana Anda belajar.....	60
A.5 Ceklis masuk M1.....	61
A.6 Yang akan Anda pelajari di M1.....	65
Bagian B: Jembatan di ArcGIS Pro.....	65
Bagian C: Jembatan di ArcMap 10.8.....	66
Lanjut belajar.....	68
Glosarium.....	69
Daftar Pustaka.....	73
Dokumentasi resmi.....	73
Aplikasi Live Penulis.....	74

Pendahuluan: Dari Klik ke Skrip: Menyuruh QGIS Bekerja Sendiri

Buku ini adalah **Seri I4** dari kelas Intermediate. Ini seri penutup kelas. Setelah ini Anda siap masuk **Masterclass GIS Modern (M1)**.

Pada seri sebelumnya, Anda memberi perintah dengan klik. Di sini Anda belajar menuliskan perintah itu sebagai **skrip**. Skrip adalah catatan perintah yang bisa dijalankan ulang, kapan saja, tanpa lelah dan tanpa lupa langkah.

Untuk siapa buku ini

Buku ini untuk Anda yang sudah membaca seri I1 sampai I3, atau paham isinya secara konsep:

- **I1:** buffer, overlay, spatial join, dissolve, dan model sederhana.
- **I2:** raster calculator, NDVI, dan statistik raster.
- **I3:** data lapangan, GeoPackage, dan pemeriksaan kualitas data.

Anda **tidak** perlu pernah memprogram. Bab 1 mengajarkan lima hal Python yang cukup untuk memulai. Konsep dari I1 sampai I3 diulang singkat di tempat yang memakainya.

Cara membaca buku ini

Satu bab terbagi dalam bagian dengan pola yang sama:

- **Studi kasus:** masalah kecil dari "KPH Contoh", kantor pengelola hutan fiktif.
- **Konsep:** inti gagasan dalam tiga kalimat, dengan analogi.
- **Langkah:** **Bagian A** untuk QGIS (acuan utama, semua skripnya sudah diuji). **Bagian B** untuk ArcGIS Pro dengan ArcPy. **Bagian C** untuk ArcMap 10.8 dengan ArcPy (Python 2.7).
- **Cek paham:** tiga pertanyaan singkat beserta jawabannya.
- **Kesalahan umum:** jebakan yang sering menimpa pemula.
- **Ringkasan, latihan, dan tabel perbandingan** tiga perangkat.

Anda tidak wajib memakai ketiganya. Pilih satu perangkat, lalu ikuti bagian yang sesuai. Bila sebuah fitur tidak ada di perangkat tertentu, buku ini menyebutnya terus terang.

Tanda [CEK]: Skrip QGIS dalam buku ini sudah dijalankan di QGIS 4.0.2 dengan data sintesis, dan hasilnya dicantumkan. Skrip ArcPy ditulis dari dokumentasi resmi dan ditandai **[CEK]**. Artinya: coba dulu pada data latihan, lalu sesuaikan dengan versi Anda.

Kenalan lagi dengan KPH Contoh

Cerita berlanjut di **KPH Contoh**. Semua nama, tempat, dan peristiwa di dalamnya fiktif. Tokohnya hanya dikenal lewat peran: **Analisis Muda** (itu Anda) dan **Kepala Seksi** (atasan yang sering memberi tugas mendadak).

Di kelas Basic, Anda menyiapkan peta dasar KPH Contoh. Di kelas Intermediate, Anda menganalisis. Di seri ini, Anda mengotomatiskan. Tiap awal bulan Kepala Seksi minta laporan yang sama: data lapangan, sempadan sungai, titik panas, dan kondisi vegetasi. Tujuan Anda: laporan itu keluar dari satu skrip, dalam hitungan detik.

Perangkat dan data

- **QGIS:** buku ini memakai QGIS 4.x. Tampilan versi lain bisa sedikit berbeda.
- **ArcGIS Pro dan ArcMap 10.8:** Bagian B dan C. Alat Spatial Analyst dipakai pada langkah raster, jadi ekstensi itu perlu aktif.
- **Paket data:** folder `paket-i4` dari penulis. Semua data berkoordinat WGS 84 / UTM zona 49S (EPSG:32749), kecuali kiriman `tim_d` (derajat) yang sengaja berbeda. Data dibuat buatan oleh skrip `buat_paket_i4.py`. Hasil analisis dari data ini **bukan** kondisi lapangan sebenarnya.

Isi paket-i4	Keterangan
Batas_KPH.gpkg	Tiga petak: KPH Alpha 4 ha, KPH Beta 4 ha, KPH Gamma 8 ha
Sungai.gpkg	Dua garis sungai (field NAMA, ORDE)
Hotspot_Nasional.gpkg	12 titik panas (field ID_Titik)
Survei.gpkg, Poligon_Sawah.gpkg	Titik survei dan poligon klasifikasi
Citra_KPH.tif	Citra 4 band (Merah, Hijau, Biru, NIR), piksel 2 m, menutup seluruh area
DSM_Drone.tif, DTM_Drone.tif	Model tinggi permukaan dan tanah, piksel 0,5 m
Ortofoto_Mentah/	Tiga ubin foto udara (band 1 = NIR, band 2 = Merah)
Ortofoto_Campur/	Tiga ubin baik dan satu ubin rusak
Kiriman_Lapangan/	Enam kiriman tim: tiga baik, satu berderajat, satu rusak, satu kekurangan field
shp/, Kiriman_Lapangan/shp/	Salinan Shapefile untuk ArcGIS Pro dan ArcMap

Menyiapkan folder

Salin folder `paket-i4` ke `D:/Latihan/paket-i4`. Buat juga folder kosong `D:/Latihan/hasil_i4` untuk keluaran. Semua skrip membaca dua baris di awalnya:

PYTHON

```
DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
```

Bila folder Anda berbeda, ubah **dua baris itu saja**. Tulis jalur dengan garis miring biasa (/). Python membaca garis miring balik (\) sebagai kode khusus.

Hubungan dengan seri lain

- **Sebelum:** I1, I2, dan I3 memberi bahan. Seri ini mengerjakan ulang bahan itu lewat skrip.

- **Sesudah:** Bab 7 adalah **Bab Jembatan ke Master**. Isinya proyek mini terpadu dan ceklis masuk **Masterclass GIS Modern (M1)**. Seri M2 dan M4 juga akan memakai kemampuan ini.

Catatan tentang ilustrasi: Gambar di buku ini adalah **skema buatan penulis**, bukan tangkapan layar. Letak tombol di komputer Anda bisa sedikit berbeda. Ikuti nama menu yang tertulis.

Kontak penulis: badarmubarok@gmail.com.

Hak cipta dan penggunaan

Hak cipta © 2026 Badar Mubarak Yogaswara. All Rights Reserved.

Hak cipta dilindungi undang-undang. Dilarang mengutip atau memperbanyak sebagian atau seluruh isi buku ini tanpa izin tertulis dari penulis. Untuk penggunaan lain, misalnya bahan ajar berbayar atau penerbitan ulang, hubungi penulis di badarmubarok@gmail.com.

Data latihan di buku ini buatan (sintetis), bukan data lapangan. Nama produk dan merek, misalnya QGIS dan ArcGIS, milik pemiliknya masing-masing.

Unduhan: buku dan paket data

Buku ini dan semua berkas latihannya tersedia gratis di situs penulis. Alamat lengkapnya tertulis di bawah, jadi bisa Anda ketik langsung bila membaca versi cetak.

Berkas	Tautan	Isi
Halaman unduhan I4	https://badarmubarok.id/buku-gis/i4/unduhan.html	semua berkas seri ini
Buku (PDF)	https://badarmubarok.id/buku-gis/i4/unduhan/BUKU-I4.pdf	buku ini dalam PDF
Paket data latihan	https://badarmubarok.id/buku-gis/i4/unduhan/data-sintetis-i4.zip	data sintetis latihan
Skrip (zip)	https://badarmubarok.id/buku-gis/i4/unduhan/skrip-i4.zip	skrip Python dan lainnya
Seluruh seri	https://badarmubarok.id/buku-gis/	12 buku Basic sampai Master

Pindai kode QR ini dengan kamera ponsel untuk membuka halaman unduhan I4.



BAB 1: Python di Dalam QGIS: Bicara dengan Peta Lewat Perintah

Studi kasus: "Daftar petak sebelum rapat"

Kepala Seksi menghampiri meja Anda lima menit sebelum rapat. "Tolong daftar petak KPH Contoh, lengkap dengan luasnya dalam hektare." Biasanya Anda membuka tabel atribut, menghitung luas, lalu mengetik ulang di lembar lain. Hari ini Anda mencoba cara baru: satu skrip pendek yang membaca layer lalu menulis daftarnya sendiri. Bulan depan, skrip yang sama bisa dijalankan lagi.

Konsep: Python di QGIS dalam tiga kalimat

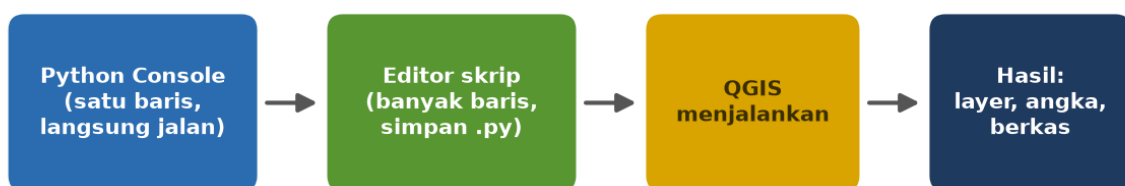
Python adalah bahasa perintah tertulis, dan **PyQGIS** adalah cara Python berbicara dengan QGIS. Hampir semua yang bisa Anda klik di QGIS punya perintah padanan di PyQGIS. Memakai menu itu seperti memesan makanan lewat pelayan, sedangkan skrip seperti daftar pesanan tertulis yang bisa diserahkan lagi setiap hari.

Istilah baru bab ini:

- **Skrip**: berkas teks berisi perintah Python, biasanya berakhiran `.py`.
- **Konsol Python**: jendela di QGIS tempat perintah diketik atau dijalankan.
- **Proyek**: proyek QGIS yang sedang terbuka, dilihat dari Python lewat `QgsProject`.
- **Layer**: satu data di proyek, bisa vektor (`QgsVectorLayer`) atau raster (`QgsRasterLayer`).
- **Fitur**: satu baris di layer vektor, yaitu satu objek peta.

Ilustrasi 1.1: Dua cara memberi perintah Python

Dua cara memberi perintah Python di QGIS



Mulai dari konsol untuk mencoba. Pindah ke skrip saat langkahnya sudah tetap.

Skema; bukan tangkapan layar

BAGIAN A | QGIS

Bagian A: PyQGIS di QGIS

A.1 Membuka Konsol Python

1. Buka QGIS, lalu pilih **Plugins ► Python Console**. Pintasannya **Ctrl+Alt+P**. Panel konsol muncul.
2. Ketik `print("Halo KPH Contoh")`, lalu tekan Enter. Tulisan Halo KPH Contoh muncul di konsol.
3. Klik tombol **Show Editor** di bilah konsol. Editor skrip terbuka di samping konsol.
4. Tempel sebuah skrip di editor. Jalankan seluruhnya dengan **Ctrl+Shift+E**. Untuk menjalankan baris yang disorot saja, tekan **Ctrl+E**.

Hasilnya: Anda punya dua tempat bekerja. Konsol cocok untuk mencoba satu baris. Editor cocok untuk skrip yang akan Anda simpan dan pakai lagi.

Modul yang sudah siap: Konsol memuat beberapa modul otomatis, termasuk `qgis.core` dan `processing`. Walau begitu, tulis `import` di awal skrip. Dengan cara itu skrip tetap jalan di tempat lain, misalnya di editor yang dijalankan dari luar QGIS.

A.2 Lima hal Python yang cukup untuk memulai

Anda tidak perlu menguasai Python untuk mengikuti buku ini. Lima hal di bawah sudah cukup. Istilah barunya: **variabel**, **daftar**, **kamus**, **perulangan**, dan **fungsi**.

- **Variabel** adalah kotak bernama yang menyimpan nilai.
- **Daftar** adalah kumpulan nilai berurutan. Nomor urutnya dimulai dari 0.
- **Kamus** adalah pasangan kunci dan nilai, seperti daftar isi berisi nomor halaman.
- **Perulangan** (`for`) mengulang perintah untuk tiap isi daftar.
- **Fungsi** adalah resep bernama yang bisa dipakai berulang kali.

Baris yang menjorok ke dalam (indentasi) menandai isi sebuah blok. Empat spasi sudah cukup. Coba skrip berikut di editor. Skrip ini belum memakai QGIS sama sekali.

PYTHON 1.0

```
# [SKRIP 1.0: Lima hal Python yang cukup untuk memulai]
# Penulis: Badar Mubarak Yogaswara
# Tidak memakai QGIS sama sekali. Boleh dicoba di Python Console baris demi baris.

# 1. Variabel: kotak bernama yang menyimpan nilai
nama = "KPH Alpha"
luas_ha = 4.0
print(nama, "luasnya", luas_ha, "ha")

# 2. Daftar: kumpulan nilai berurutan. Nomor urut dimulai dari 0, bukan 1.
petak = ["KPH Alpha", "KPH Beta", "KPH Gamma"]
print("Petak pertama:", petak[0])
print("Jumlah petak:", len(petak))

# 3. Kamus: pasangan kunci dan nilai, seperti daftar isi dengan nomor halaman
luas = {"KPH Alpha": 4.0, "KPH Beta": 4.0, "KPH Gamma": 8.0}
print("Luas KPH Gamma:", luas["KPH Gamma"])
```

```
# 4. Perulangan for dan percabangan if. Baris yang menjorok ke dalam ikut dalam blok.
total = 0
for nama, ha in luas.items():
    total = total + ha
    if ha > 5:
        print(nama, "termasuk petak besar")
print("Total:", total, "ha")

# 5. Fungsi: resep yang diberi nama supaya bisa dipakai berulang kali
def ke_hektar(meter_persegi):
    return meter_persegi / 10000

print("40000 m2 =", ke_hektar(40000), "ha")
```

Hasil uji di QGIS 4.0.2:

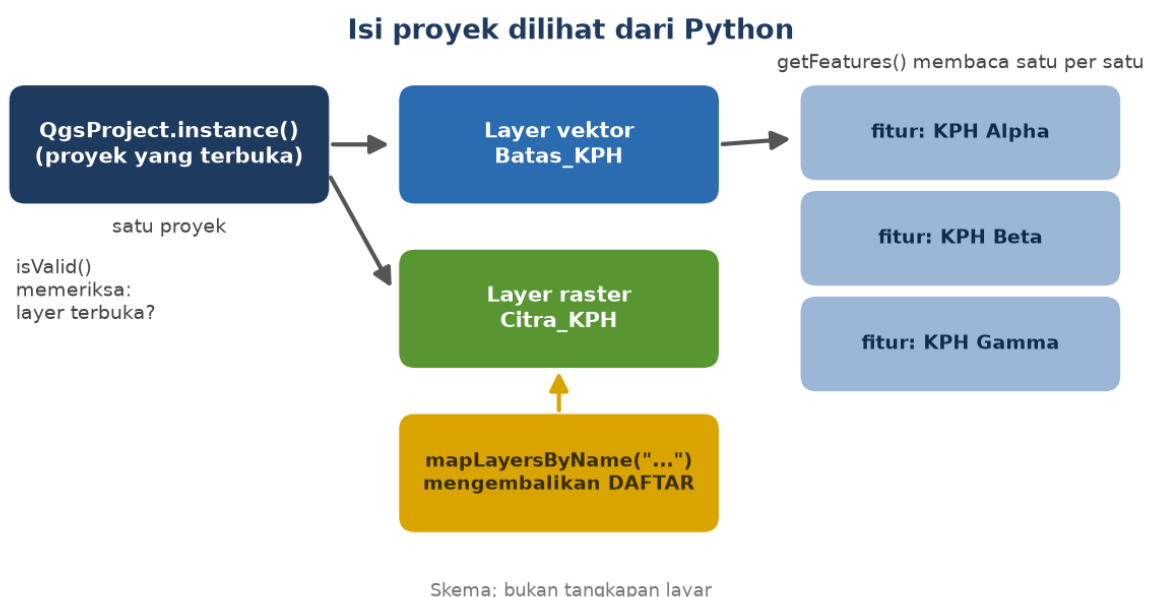
HASIL UJI SKRIP 1.0

```
KPH Alpha luasnya 4.0 ha
Petak pertama: KPH Alpha
Jumlah petak: 3
Luas KPH Gamma: 8.0
KPH Gamma termasuk petak besar
Total: 16.0 ha
40000 m2 = 4.0 ha
```

A.3 Memuat layer dan mengambilnya dengan nama

Bayangkan rak buku. Anda meletakkan buku di rak (memuat layer), lalu mengambilnya lagi dengan menyebut judulnya (mengambil layer dengan nama). Rak itu adalah proyek.

Ilustrasi 1.2: Isi proyek dilihat dari Python



Skrip 1.1 memuat satu layer vektor dan satu layer raster. Lalu skrip mengambil keduanya kembali dan mencetak sifatnya.

PYTHON (PYQGIS) 1.1

```
# [SKRIP 1.1: Memuat layer lewat Python dan mengambilnya kembali dengan nama]
# Penulis: Badar Mubarak Yogaswara
# Jalankan di Plugins > Python Console. Ganti DATA dengan folder paket-i4 di komputer Anda.
from qgis.core import QgsProject, QgsVectorLayer, QgsRasterLayer, QgsWkbTypes

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

proyek = QgsProject.instance()          # proyek yang sedang terbuka
proyek.removeAllMapLayers()             # kosongkan dulu agar tidak dobel

# 1. Muat satu layer vektor dan satu raster. Argumen: jalur, nama di panel Layers, penyedia data.
batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
dsm = QgsRasterLayer(DATA + "/DSM_Drone.tif", "DSM_Drone", "gdal")

# 2. Periksa dulu: layer yang gagal dibuka tidak memicu galat, hanya isValid() bernilai False.
print("Batas_KPH sah?", batas.isValid())
print("DSM_Drone sah?", dsm.isValid())

# 3. Masukkan ke proyek supaya tampil di panel Layers.
proyek.addMapLayer(batas)
proyek.addMapLayer(dsm)

# 4. Ambil layer dengan namanya. mapLayersByName() mengembalikan DAFTAR, ambil isi pertama.
layer = proyek.mapLayersByName("Batas_KPH")[0]
print("Nama:", layer.name())
print("CRS:", layer.crs().authid())
print("Jenis geometri:", QgsWkbTypes.displayString(layer.wkbType()))
print("Jumlah fitur:", layer.featureCount())
print("Nama field:", [f.name() for f in layer.fields()])

# 5. Sifat sebuah raster
r = proyek.mapLayersByName("DSM_Drone")[0]
print("Raster:", r.width(), "x", r.height(), "piksel", "r.bandCount(), "band")
print("Ukuran piksel (m):", r.rasterUnitsPerPixelX())
print("Jumlah layer di proyek:", len(proyek.mapLayers()))
```

Hasil uji di QGIS 4.0.2:**HASIL UJI SKRIP 1.1**

```
Batas_KPH sah? True
DSM_Drone sah? True
Nama: Batas_KPH
CRS: EPSG:32749
Jenis geometri: Polygon
Jumlah fitur: 3
Nama field: ['fid', 'NAMA_KPH']
Raster: 800 x 800 piksel, 1 band
Ukuran piksel (m): 0.5
Jumlah layer di proyek: 2
```

Bedah kode: `QgsVectorLayer(jalur, nama, "ogr")` membuat layer vektor. Kata "ogr" adalah penyedia data untuk vektor. Layer raster memakai "gdal". Nama kedua dipakai sebagai nama di panel Layers. `isValid()` menjawab benar atau salah. Layer yang gagal dibuka **tidak** menimbulkan pesan galat, jadi periksa sendiri. `mapLayersByName("Batas_KPH")` mengembalikan **daftar**, sebab bisa ada beberapa layer bernama sama. Karena itu ditulis `[0]` untuk mengambil yang pertama.

A.4 Membaca fitur dan atribut

Layer vektor seperti tabel. Tiap baris adalah satu fitur. Skrip 1.2 membaca tiap baris, menghitung luas, memilih satu petak dengan ekspresi, lalu menulis hasil ke berkas CSV.

PYTHON (PYQGIS) 1.2

```
# [SKRIP 1.2: Membaca fitur dan atribut, memilih dengan ekspresi, menulis CSV]
# Penulis: Badar Mubarak Yogaswara

import csv
import os
from qgis.core import QgsProject, QgsVectorLayer

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
os.makedirs(HASIL, exist_ok=True)

batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
QgsProject.instance().addMapLayer(batas)

# 1. Satu fitur = satu baris tabel atribut. getFeatures() membacanya satu per satu.
baris = []
for fitur in batas.getFeatures():
    nama = fitur["NAMA_KPH"]
    luas_ha = fitur.geometry().area() / 10000      # m2 menjadi hektare (CRS harus berupa meter)
    baris.append((nama, round(luas_ha, 2)))
    print(nama, round(luas_ha, 2), "ha")

total = sum(luas for _, luas in baris)
print("Total:", total, "ha")

# 2. Memilih fitur dengan ekspresi, sama seperti Select by Expression
batas.selectByExpression('"NAMA_KPH" = \'KPH Beta\'')
print("Terpilih:", batas.selectedFeatureCount(), "fitur")
for fitur in batas.selectedFeatures():
    print("  ->", fitur["NAMA_KPH"])
batas.removeSelection()

# 3. Menulis hasil ke berkas CSV
jalur = HASIL + "/luas_kph.csv"
with open(jalur, "w", newline="", encoding="utf-8") as f:
    penulis = csv.writer(f)
    penulis.writerow(["NAMA_KPH", "LUAS_HA"])
    penulis.writerows(baris)
print("CSV tersimpan:", os.path.exists(jalur))
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 1.2

KPH Alpha 4.0 ha
KPH Beta 4.0 ha
KPH Gamma 8.0 ha

```
Total: 16.0 ha
Terpilih: 1 fitur
-> KPH Beta
CSV tersimpan: True
```

Angka luas tepat karena layer berkoordinat meter (UTM). Pada layer berderajat (EPSG:4326), `geometry().area()` menghasilkan derajat persegi, bukan meter persegi. Ingat pelajaran CRS dari seri B1.

A.5 Menyimpan proyek dan membukanya lagi

Skrip 1.3 menyusun proyek dari nol: mengatur CRS, memuat empat layer, menyimpan sebagai `.qgz`, mengosongkan proyek, lalu membukanya kembali sebagai bukti.

PYTHON (PYQGIS) 1.3

```
# [SKRIP 1.3: Menyusun proyek lewat Python lalu menyimpan dan membukanya lagi]
# Penulis: Badar Mubarak Yogaswara
import os
from qgis.core import QgsProject, QgsVectorLayer, QgsRasterLayer, QgsCoordinateReferenceSystem

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
os.makedirs(HASIL, exist_ok=True)

proyek = QgsProject.instance()
proyek.removeAllMapLayers()
proyek.setCrs(QgsCoordinateReferenceSystem("EPSG:32749"))      # CRS proyek: UTM zona 49S
proyek.setTitle("KPH Contoh - proyek skrip")

daftar = [
    QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr"),
    QgsVectorLayer(DATA + "/Sungai.gpkg", "Sungai", "ogr"),
    QgsVectorLayer(DATA + "/Hotspot_Nasional.gpkg", "Hotspot_Nasional", "ogr"),
    QgsRasterLayer(DATA + "/Citra_KPH.tif", "Citra_KPH", "gdal"),
]
for layer in daftar:
    if layer.isValid():
        proyek.addMapLayer(layer)
    else:
        print("GAGAL dimuat:", layer.name())

berkas = HASIL + "/Proyek_KPH.qgz"
print("Tersimpan?", proyek.write(berkas))      # write() mengembalikan True bila berhasil

# Buka kembali dan buktikan isinya utuh
proyek.clear()
print("Setelah dikosongkan:", len(proyek.mapLayers()), "layer")
print("Terbuka?", proyek.read(berkas))
print("Judul:", proyek.title(), "| CRS:", proyek.crs().authid())
print("Layer:", sorted(l.name() for l in proyek.mapLayers().values()))
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 1.3

```
Tersimpan? True
Setelah dikosongkan: 0 layer
Terbuka? True
Judul: KPH Contoh - proyek skrip | CRS: EPSG:32749
Layer: ['Batas_KPH', 'Citra_KPH', 'Hotspot_Nasional', 'Sungai']
```

Hasilnya: proyek yang dibuat lewat skrip sama utuhnya dengan proyek yang dibuat lewat klik.

BAGIAN B | ArcGIS Pro

Bagian B: ArcPy dasar di ArcGIS Pro

ArcPy adalah paket Python milik Esri. ArcGIS Pro memakai Python 3. Jendela perintahnya dibuka lewat tab **Analysis ► Python ► Python Window**. [CEK]

Padanannya dengan Bagian A: `arcpy.ListFeatureClasses()` untuk daftar data, `arcpy.da.SearchCursor` untuk membaca baris, `arcpy.Raster` untuk raster, dan `arcpy.mp.ArcGISProject("CURRENT")` untuk proyek yang terbuka. Bacalah skrip berikut dengan membandingkannya dengan Skrip 1.1 sampai 1.3.

PYTHON (ARCPY, PYTHON 3)

```
# [SKRIP 1.1-Pro: ArcPy dasar: membaca data, atribut, dan proyek]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi ArcGIS Pro Anda.
import arcpy

DATA = r"D:\Latihan\paket-i4"
arcpy.env.workspace = DATA + r"\shp"          # folder Shapefile; daftar data dibaca dari sini

# 1. Daftar data dan sifatnya
print(arcpy.ListFeatureClasses())              # contoh: ['Batas_KPH.shp', 'Hotspot_Nasional.shp', ...]
batas = "Batas_KPH.shp"
d = arcpy.Describe(batas)
print(d.shapeType, d.spatialReference.name, d.spatialReference.factoryCode)  # Polygon, ..., 32749
print("Jumlah fitur:", int(arcpy.management.GetCount(batas)[0]))
print("Field:", [f.name for f in arcpy.ListFields(batas)])

# 2. Membaca tiap baris dengan kursor (padanan getFeatures() di QGIS)
with arcpy.da.SearchCursor(batas, ["NAMA_KPH", "SHAPE@AREA"]) as kursor:
    for nama, luas in kursor:
        print(nama, round(luas / 10000, 2), "ha")          # luas dalam satuan peta (m2) dibagi 10.000

# 3. Raster
r = arcpy.Raster(DATA + r"\Citra_KPH.tif")
print("Raster:", r.width, "x", r.height, "piksel,", r.bandCount, "band; piksel", r.meanCellWidth, "m")

# 4. Proyek: tambah data lalu ambil layer dengan nama
aprx = arcpy.mp.ArcGISProject("CURRENT")        # proyek yang sedang terbuka
peta = aprx.listMaps()[0]
peta.addDataFromPath(DATA + r"\shp\Batas_KPH.shp")
layer = peta.listLayers("Batas_KPH")[0]         # listLayers() juga mengembalikan DAFTAR
print("Layer ditemukan:", layer.name)
aprx.save()
```

Perlu dicek: Skrip ini perlu Anda cocokkan dengan ArcGIS Pro Anda. Data yang dibaca adalah Shapefile di folder `shp`. Cocokkan dengan versi Anda. [CEK]

Beda yang perlu diingat. `listLayers("Batas_KPH")` juga mengembalikan daftar, sama seperti `mapLayersByName`. `Cursor SearchCursor` membaca nilai lewat urutan nama field yang Anda sebut, bukan lewat nama di tiap baris.

BAGIAN C | ArcMap 10.8

Bagian C: ArcPy dasar di ArcMap 10.8

ArcMap memakai **Python 2.7**. Jendela perintahnya dibuka lewat menu **Geoprocessing ► Python**. [CEK] Pengelolaan dokumen peta memakai modul `arcpy.mapping`, bukan `arcpy.mp`.

Dua beda penting dengan Python 3. Pertama, pembagian dua bilangan bulat menghasilkan bilangan bulat, jadi tulis `10000.0` agar hasilnya desimal. Kedua, tidak ada f-string, jadi teks disusun dengan tanda `%`.

PYTHON (ARCPY, PYTHON 2.7)

```
# -*- coding: utf-8 -*-
# [SKRIP 1.1-ArcMap: ArcPy dasar: membaca data, atribut, dan dokumen peta (Python 2.7)]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi ArcMap Anda.
import arcpy

DATA = r"D:\Latihan\paket-i4"
arcpy.env.workspace = DATA + r"\shp"

# 1. Daftar data dan sifatnya
print(arcpy.ListFeatureClasses())
batas = "Batas_KPH.shp"
d = arcpy.Describe(batas)
print("%s %s %s" % (d.shapeType, d.spatialReference.name, d.spatialReference.factoryCode))
print("Jumlah fitur: %d" % int(arcpy.GetCount_management(batas).getOutput(0)))
print("Field: %s" % [f.name for f in arcpy.ListFields(batas)])

# 2. Membaca tiap baris dengan kursor
with arcpy.da.SearchCursor(batas, ["NAMA_KPH", "SHAPE@AREA"]) as kursor:
    for nama, luas in kursor:
        print("%s %.2f ha" % (nama, luas / 10000.0))          # 10000.0: hasil bagi desimal di Python 2

# 3. Raster
r = arcpy.Raster(DATA + r"\Citra_KPH.tif")
print("Raster: %d x %d piksel, %d band, piksel %s m" % (r.width, r.height, r.bandCount, r.meanCellWidth))

# 4. Dokumen peta: tambah layer lalu ambil dengan nama
mxd = arcpy.mapping.MapDocument("CURRENT")
df = arcpy.mapping.ListDataFrames(mxd)[0]
arcpy.mapping.AddLayer(df, arcpy.mapping.Layer(DATA + r"\shp\Batas_KPH.shp"))
layer = arcpy.mapping.ListLayers(mxd, "Batas_KPH", df)[0]    # juga mengembalikan DAFTAR
print("Layer ditemukan: %s" % layer.name)
mxd.save()
```

Perlu dicek: Skrip ini perlu Anda cocokkan dengan ArcMap Anda. ArcMap 10.8 membaca Shapefile dengan baik. Dukungan GeoPackage di ArcMap tergantung versi, jadi skrip memakai Shapefile. [CEK]

Cek paham

1. Mengapa `mapLayersByName("Batas_KPH")` harus diikuti `[0]`?
2. Apa beda "ogr" dan "gdal" saat membuat layer?
3. Berkas data hilang, tetapi skrip tidak menampilkan galat. Bagaimana cara tahu bahwa layer gagal dibuka?

Jawaban:

1. Karena hasilnya berupa daftar. Bisa ada beberapa layer dengan nama sama. `[0]` mengambil yang pertama.
2. "ogr" untuk data vektor, "gdal" untuk raster.
3. Periksa `layer.isValid()`. Nilai `False` berarti layer tidak terbuka.

Kesalahan umum

- **Memakai garis miring balik di jalur.** Tulis `D:/Latihan/paket-i4`, bukan `D:\Latihan\paket-i4`. Bila ingin garis miring balik,awali jalur dengan `r` seperti `r"D:\Latihan"`.
- **Menjalankan skrip dua kali lalu muncul layer ganda.** Kosongkan proyek lebih dulu dengan `removeAllMapLayers()`, seperti di Skrip 1.1.
- **Salah huruf pada nama layer.** "batas_kph" berbeda dari "Batas_KPH". Hasilnya daftar kosong, lalu muncul galat `IndexError` saat memanggil `[0]`.
- **Indentasi tidak rapi.** Baris dalam blok `for` harus sama menjoroknya.

Ringkasan dan latihan

Ringkasan: skrip adalah catatan perintah yang bisa diulang. `QgsProject.instance()` adalah proyek yang terbuka. Layer dimuat dengan `QgsVectorLayer` atau `QgsRasterLayer`, diperiksa dengan `isValid()`, lalu diambil dengan `mapLayersByName()[0]`. Tiap baris vektor dibaca dengan `getFeatures()`.

Latihan:

1. Ubah Skrip 1.2 agar hanya mencetak petak yang luasnya lebih dari 5 ha. Hasil yang benar: hanya KPH Gamma.
2. Muat `Sungai.gpkg`, lalu cetak panjang tiap sungai dengan `fitur.geometry().length()`. Hasil yang benar: Sungai Tengah 420,3 m dan Anak Sungai Timur 289,5 m, total 709,8 m.

Tabel perbandingan: PyQGIS dan ArcPy dasar

Hal	QGIS (Python 3)	ArcGIS Pro (Python 3)	ArcMap 10.8 (Python 2.7)
Jendela perintah	Plugins ► Python Console	Analysis ► Python ► Python Window [CEK]	Geoprocessing ► Python [CEK]
Memuat data ke peta	<code>QgsVectorLayer</code> lalu <code>addMapLayer</code>	<code>Map.addDataFromPath</code> [CEK]	<code>arcpy.mapping.AddLayer</code> [CEK]
Ambil layer dengan nama	<code>mapLayersByName("x")[0]</code>	<code>Map.listLayers("x")[0]</code> [CEK]	<code>ListLayers(mxd, "x", df)[0]</code> [CEK]
Baca tiap baris	<code>getFeatures()</code>	<code>arcpy.da.SearchCursor</code> [CEK]	<code>arcpy.da.SearchCursor</code> [CEK]
Simpan proyek	<code>QgsProject.write()</code>	<code>aprx.save()</code> [CEK]	<code>mxd.save()</code> [CEK]
Cek layer sah	<code>isValid()</code>	<code>arcpy.Exists()</code> [CEK]	<code>arcpy.Exists()</code> [CEK]

BAB 2: Alat Processing dari Skrip: Satu Perintah, Satu Analisis

Studi kasus: "Sempadan sungai tiap petak"

Kepala Seksi ingin tahu berapa hektare jalur sempadan sungai di tiap petak, dan berapa titik panas di dalamnya. Untuk latihan, lebar sempadan ditetapkan 30 meter. Di seri I1, Anda menjalankan Buffer lalu Intersection satu per satu dengan klik. Kini seluruh rangkaian itu Anda tulis sekali, lalu dijalankan dengan satu tombol.

Konsep: alat Processing dalam tiga kalimat

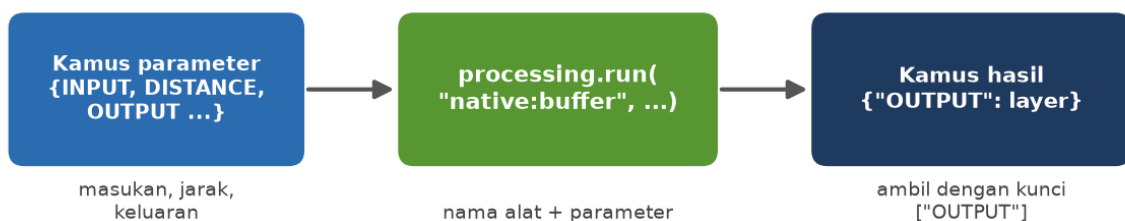
Setiap alat di Processing Toolbox bisa dipanggil dari Python dengan `processing.run`. Anda memberi **ID alat** dan **kamus parameter**, lalu QGIS mengembalikan **kamus hasil**. Ini seperti mengisi formulir pesanan: menu memberi kolom kosong, sedangkan skrip mengisinya dengan tulisan.

Istilah baru bab ini:

- **ID alat**: nama unik alat, misalnya `native:buffer`.
- **Parameter**: isian alat, seperti masukan, jarak, dan keluaran.
- **Keluaran sementara** (`TEMPORARY_OUTPUT`): hasil yang disimpan di memori atau folder sementara, bukan di berkas pilihan Anda.
- **Rantai alat**: keluaran satu alat menjadi masukan alat berikutnya.
- **Statistik zonal**: ringkasan nilai raster di dalam tiap poligon.

Ilustrasi 2.1: Satu panggilan, kamus masuk dan kamus keluar

Satu panggilan: kamus masuk, kamus keluar



`OUTPUT = "TEMPORARY_OUTPUT"`: vektor kembali sebagai layer, raster kembali sebagai jalur berkas (teks)

Skema; bukan tangkapan layar

BAGIAN A | QGIS

Bagian A: processing.run di QGIS

A.1 Menemukan ID alat dan parameternya

Pertanyaan pertama: apa nama alat dan parameternya? Skrip 2.1 menjawabnya dengan Python.

Bagian pertamanya mencari ID alat dengan kata kunci. Bagian keduanya mencetak nama parameter alat Buffer. Skrip lalu menjalankan Buffer pada layer sungai.

Cara lain yang tersedia: `processing.algorithmHelp("native:buffer")` mencetak bantuan lengkap alat itu, termasuk nilai yang diterima tiap parameter. Panel **Processing ► History** juga mencatat tiap alat yang pernah Anda jalankan lewat dialog. [CEK: bentuk catatannya]

PYTHON (PYQGIS) 2.1

```
# [SKRIP 2.1: Menjalankan satu alat Processing dari Python (Buffer)]
# Penulis: Badar Mubarak Yogaswara
import processing
from qgis.core import QgsProject, QgsVectorLayer, QgsApplication

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

sungai = QgsVectorLayer(DATA + "/Sungai.gpkg", "Sungai", "ogr")

# 0. Mencari ID alat dengan kata kunci
for a in sorted(QgsApplication.processingRegistry().algorithms(), key=lambda x: x.id()):
    if a.id().startswith("native:") and "buffer" in a.id():
        print("ID:", a.id(), "-", a.displayName())

# 1. Mengintip "formulir" alat: nama parameter dan artinya
alat = QgsApplication.processingRegistry().algorithmById("native:buffer")
print("Alat:", alat.displayName())
for p in alat.parameterDefinitions():
    print("  ", p.name(), "=", p.description())

# 2. Menjalankan alat. Parameter berupa kamus (dictionary): nama parameter -> nilai.
hasil = processing.run("native:buffer", {
    "INPUT": sungai,
    "DISTANCE": 30,                # satuan meter karena CRS-nya UTM
    "SEGMENTS": 8,
    "END_CAP_STYLE": 0,            # 0 = bulat
    "JOIN_STYLE": 0,              # 0 = bulat
    "MITER_LIMIT": 2,
    "DISSOLVE": True,             # gabungkan semua penyangga jadi satu bentuk
    "OUTPUT": "TEMPORARY_OUTPUT", # keluaran sementara, tidak menulis berkas
})

# 3. Hasilnya juga kamus. Kuncinya nama keluaran. Untuk vektor sementara, isinya sudah berupa layer.
sempadan = hasil["OUTPUT"]
print("Jenis keluaran:", type(sempadan).__name__)
print("Fitur:", sempadan.featureCount())
luas = sum(f.geometry().area() for f in sempadan.getFeatures()) / 10000
print("Luas sempadan sungai:", round(luas, 2), "ha")

# 4. Beri nama yang jelas lalu tampilkan di proyek
sempadan.setName("Sempadan_30m")
QgsProject.instance().addMapLayer(sempadan)
print("Layer di proyek:", [l.name() for l in QgsProject.instance().mapLayers().values()])
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 2.1

```
ID: native:buffer - Buffer
ID: native:bufferbym - Variable width buffer (by M value)
ID: native:multiringconstantbuffer - Multi-ring buffer (constant distance)
ID: native:singlesidedbuffer - Single sided buffer
ID: native:taperedbuffer - Tapered buffers
ID: native:wedgebuffers - Create wedge buffers
Alat: Buffer
  INPUT = Input layer
  DISTANCE = Distance
  SEGMENTS = Segments
  END_CAP_STYLE = End cap style
  JOIN_STYLE = Join style
  MITER_LIMIT = Miter limit
  DISSOLVE = Dissolve result
  SEPARATE_DISJOINT = Keep disjoint results separate
  OUTPUT = Buffered
Jenis keluaran: QgsVectorLayer
Fitur: 1
Luas sempadan sungai: 4.48 ha
Layer di proyek: ['Sempadan_30m']
```

Bedah kode: `processing.run("native:buffer", {...})` menerima dua hal: ID alat dan kamus parameter. Kuncinya adalah **nama parameter** (huruf besar), seperti `INPUT` dan `DISTANCE`. Satuan `DISTANCE` mengikuti CRS layer. Pada layer UTM berarti meter. Pada layer berderajat berarti derajat, jadi 30 akan menjadi 30 derajat. Pastikan CRS metrik lebih dulu. Angka seperti `END_CAP_STYLE: 0` adalah pilihan bernomor. Nomor yang berlaku tercantum pada `algorithmHelp`.

A.2 Keluaran sementara: layer atau teks?

Isian `OUTPUT` menentukan ke mana hasil dikirim. Isi dengan jalur berkas untuk menyimpan permanen. Isi dengan `"TEMPORARY_OUTPUT"` bila hasil hanya perantara. Hasil uji di QGIS 4.0.2 menunjukkan perbedaan ini:

Jenis keluaran	Isi hasil["OUTPUT"]	Cara memakainya
Vektor, <code>TEMPORARY_OUTPUT</code>	Layer (<code>QgsVectorLayer</code>) bernama "Buffered"	Langsung dipakai sebagai masukan alat lain
Raster, <code>TEMPORARY_OUTPUT</code>	Teks jalur berkas sementara	Bungkus dengan <code>QgsRasterLayer(jalur, nama, "gdal")</code>
Vektor, <code>"memory:nama"</code>	Layer	Sama seperti vektor di atas
Jalur berkas	Teks jalur berkas itu	Muat sebagai layer bila perlu

Layer sementara **tidak** muncul di panel Layers sebelum Anda memanggil `addMapLayer`. Layer sementara juga hilang saat QGIS ditutup. Simpan hasil penting ke `GeoPackage` (Bab 7 mencontohkannya).

A.3 Rantai alat vektor

Sempadan per petak butuh tiga alat. **Buffer** membuat jalur 30 m di tepi sungai. **Intersection** memotong jalur itu per petak. **Count points in polygon** menghitung titik panas. Keluaran alat pertama langsung menjadi masukan alat kedua, tanpa berkas perantara.

Ilustrasi 2.2: Rantai alat

Rantai alat: keluaran satu alat menjadi masukan alat berikutnya



Hasil antara tidak perlu disimpan: cukup keluaran sementara

Skema; bukan tangkapan layar

PYTHON (PYQGIS) 2.2

```

# [SKRIP 2.2: Rantai alat vektor: sempadan sungai dan titik panas per KPH]
# Penulis: Badar Mubarak Yogaswara
import processing
from qgis.core import QgsProject, QgsVectorLayer

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
sungai = QgsVectorLayer(DATA + "/Sungai.gpkg", "Sungai", "ogr")
hotspot = QgsVectorLayer(DATA + "/Hotspot_Nasional.gpkg", "Hotspot_Nasional", "ogr")

# Langkah 1: penyangga (buffer) 30 m di kiri-kanan sungai
sempadan = processing.run("native:buffer", {
    "INPUT": sungai, "DISTANCE": 30, "SEGMENTS": 8, "END_CAP_STYLE": 0, "JOIN_STYLE": 0,
    "MITER_LIMIT": 2, "DISSOLVE": True, "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]

# Langkah 2: irisan (intersection). Hasil langkah 1 langsung menjadi masukan langkah 2.
irisan = processing.run("native:intersection", {
    "INPUT": batas, "OVERLAY": sempadan, "INPUT_FIELDS": ["NAMA_KPH"],
    "OVERLAY_FIELDS": [], "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]

# Langkah 3: hitung titik panas di tiap KPH
dengan_titik = processing.run("native:countpointsinpolygon", {
    "POLYGONS": batas, "POINTS": hotspot, "FIELD": "JML_HOTSPOT", "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]

# Langkah 4: gabungkan angkanya dalam satu tabel kecil
luas_sempadan = {}
for f in irisan.getFeatures():
    nama = f["NAMA_KPH"]
    luas_sempadan[nama] = luas_sempadan.get(nama, 0) + f.geometry().area() / 10000

print("KPH          luas(ha)  sempadan(ha)  persen  titik_panas")
for f in dengan_titik.getFeatures():
    nama = f["NAMA_KPH"]
    luas = f.geometry().area() / 10000
    sem = luas_sempadan.get(nama, 0)
  
```

```
print("%-11s %8.2f %13.2f %7.1f %12d" % (nama, luas, sem, 100 * sem / luas, f["JML_HOTSPOT"]))
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 2.2

KPH	luas (ha)	sempadan (ha)	persen	titik_panas
KPH Alpha	4.00	1.18	29.4	2
KPH Beta	4.00	1.39	34.6	5
KPH Gamma	8.00	1.50	18.7	5

Angka dapat dicek tanpa komputer: KPH Alpha dan Beta masing-masing berluas 4 ha, dan KPH Gamma 8 ha. Jumlah titik panas 2, 5, dan 5 sama dengan 12 titik di `Hotspot_Nasional`. Total sempadan di tiga petak ($1,18 + 1,39 + 1,50 = 4,07$ ha) lebih kecil dari 4,48 ha pada Skrip 2.1, karena sebagian jalur buffer jatuh di luar batas area.

A.4 Alat raster: NDVI, tinggi tajuk, dan statistik zonal

NDVI adalah angka kehijauan: $(\text{NIR} - \text{Merah}) \div (\text{NIR} + \text{Merah})$. **CHM** adalah tinggi tajuk, yaitu DSM dikurangi DTM. Keduanya dihitung dengan `native:rastercalc`. Dalam ekspresinya, band ditulis "`NamaLayer@nomorBand`". Karena itu nama layer pada `QgsRasterLayer(...)` penting.

Setelah itu `native:zonalstatisticsfb` menghitung rata-rata nilai raster di dalam tiap poligon KPH. Pilihan statistik berupa angka: 0 hitungan, 1 jumlah, 2 rata-rata, dan seterusnya (daftar ada di `algorithmHelp`).

PYTHON (PYQGIS) 2.3

```
# [SKRIP 2.3: Alat raster dari Python: NDVI, tinggi tajuk (CHM), dan statistik per KPH]
# Penulis: Badar Mubarak Yogaswara
import processing
from qgis.core import QgsProject, QgsVectorLayer, QgsRasterLayer

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
citra = QgsRasterLayer(DATA + "/Citra_KPH.tif", "Citra_KPH", "gdal") # band 1=Merah, 4=NIR
dsm = QgsRasterLayer(DATA + "/DSM_Drone.tif", "DSM", "gdal")
dtm = QgsRasterLayer(DATA + "/DTM_Drone.tif", "DTM", "gdal")

# 1. NDVI = (NIR - Merah) / (NIR + Merah). Dalam ekspresi, band ditulis "NamaLayer@nomorBand".
jalur_ndvi = processing.run("native:rastercalc", {
    "EXPRESSION": '("Citra_KPH@4" - "Citra_KPH@1") / ("Citra_KPH@4" + "Citra_KPH@1")',
    "LAYERS": [citra], "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
print("Keluaran raster sementara berupa:", type(jalur_ndvi).__name__) # teks jalur, bukan layer
ndvi = QgsRasterLayer(jalur_ndvi, "NDVI_KPH", "gdal") # bungkus jadi layer
print("NDVI sah?", ndvi.isValid())

# 2. CHM = DSM - DTM
jalur_chm = processing.run("native:rastercalc", {
    "EXPRESSION": '"DSM@1" - "DTM@1"', "LAYERS": [dsm, dtm], "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
chm = QgsRasterLayer(jalur_chm, "CHM", "gdal")

# 3. Statistik zonal: rata-rata nilai raster di dalam tiap poligon KPH
def rata_rata(raster, awalan):
    return processing.run("native:zonalstatisticsfb", {
        "INPUT": batas, "INPUT_RASTER": raster, "RASTER_BAND": 1,
        "COLUMN_PREFIX": awalan, "STATISTICS": [2], # 2 = Mean
```

```

"OUTPUT": "TEMPORARY_OUTPUT")) ["OUTPUT"]

zn = rata_rata(ndvi, "ndvi_")
zc = rata_rata(chm, "chm_")
chm_per_kph = {f["NAMA_KPH"]: f["chm_mean"] for f in zc.getFeatures()}
print("KPH          NDVI rata-rata    CHM rata-rata (m)")
for f in zn.getFeatures():
    print("%-11s %12.3f %16.2f" % (f["NAMA_KPH"], f["ndvi_mean"], chm_per_kph[f["NAMA_KPH"]]))

QgsProject.instance().addMapLayer(ndvi)
QgsProject.instance().addMapLayer(chm)

```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 2.3

```

Keluaran raster sementara berupa: str
NDVI sah? True
KPH          NDVI rata-rata    CHM rata-rata (m)
KPH Alpha          0.604          2.60
KPH Beta           0.431          2.69
KPH Gamma          0.329          2.56

```

Angka NDVI cocok dengan hitungan tangan dari data sintetis: Alpha memakai Merah 40 dan NIR 160, jadi $(160 - 40) \div (160 + 40) = 0,60$. Beta memakai Merah 60 dan NIR 150, jadi 0,43. Gamma terdiri atas dua separuh (sekitar 0,52 dan 0,13) sehingga rata-ratanya 0,33.

Alat yang mirip, hasil berbeda: Contoh lama di internet sering memakai ID `qgis:rastercalculator`. Pada uji di QGIS 4.0.2, alat itu gagal dengan pesan perhitungan saat dipanggil seperti Skrip 2.3. Pakai `native:rastercalc` yang terdokumentasi dan sudah diuji di sini.

BAGIAN B | ArcGIS Pro

Bagian B: Alat geoprocessing lewat ArcPy di ArcGIS Pro

Di ArcPy, setiap alat adalah fungsi: `arcpy.analysis.Buffer`, `arcpy.analysis.Intersect`, dan `arcpy.analysis.SpatialJoin`. Parameter ditulis berurutan atau dengan nama. Hasil antara disimpan di **ruang kerja memory**, misalnya `memory\sempadan`. Ruang ini padanan `TEMPORARY_OUTPUT`: cepat, tetapi memakai RAM dan hilang saat Pro ditutup.

Beda penting dengan QGIS. Menghitung titik per petak memakai **Spatial Join** (kolom `Join_Count` otomatis terbentuk). Statistik zonal memakai `ZonalStatisticsAsTable` dari `Spatial Analyst`. NDVI memakai `Float` dan band ditulis sebagai `jalur\Band_n`. [CEK]

PYTHON (ARCPY, PYTHON 3)

```

# [SKRIP 2.1-Pro: Rantai alat: sempadan sungai, titik panas, dan NDVI per KPH dengan ArcPy]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi Anda. Zonal Statistics as Table dan Float butuh ekstensi Spatial Analyst.
import arcpy
from arcpy.sa import Raster, Float, ZonalStatisticsAsTable

DATA = r"D:\Latihan\paket-i4"

```

```
HASIL = r"D:\Latihan\hasil_i4"
arcpy.env.workspace = DATA + r"\shp"
arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")

# Keluaran sementara: ruang kerja "memory" (padanan TEMPORARY_OUTPUT di QGIS)
arcpy.analysis.Buffer("Sungai.shp", r"memory\sempadan", "30 Meters", dissolve_option="ALL")
arcpy.analysis.Intersect(["Batas_KPH.shp", r"memory\sempadan"], r"memory\irisan")
arcpy.analysis.SpatialJoin("Batas_KPH.shp", "Hotspot_Nasional.shp", r"memory\kph_titik",
                           "JOIN_ONE_TO_ONE", "KEEP_ALL", match_option="INTERSECT") # menambah Join_Count

sempadan = {}
with arcpy.da.SearchCursor(r"memory\irisan", ["NAMA_KPH", "SHAPE@AREA"]) as k:
    for nama, luas in k:
        sempadan[nama] = sempadan.get(nama, 0) + luas / 10000
with arcpy.da.SearchCursor(r"memory\kph_titik", ["NAMA_KPH", "Join_Count", "SHAPE@AREA"]) as k:
    for nama, jml, luas in k:
        print("%s: luas %.2f ha, sempadan %.2f ha, titik panas %d"
              % (nama, luas / 10000, sempadan.get(nama, 0), jml))

# NDVI = (NIR - Merah) / (NIR + Merah); band 4 = NIR, band 1 = Merah. Nama band ditulis jalur\Band_n.
nir = Float(Raster(DATA + r"\Citra_KPH.tif\Band_4"))
merah = Float(Raster(DATA + r"\Citra_KPH.tif\Band_1"))
ndvi = (nir - merah) / (nir + merah)
ndvi.save(HASIL + r"\NDVI_KPH.tif")
ZonalStatisticsAsTable("Batas_KPH.shp", "NAMA_KPH", ndvi, r"memory\ndvi_zonal", "DATA", "MEAN")
with arcpy.da.SearchCursor(r"memory\ndvi_zonal", ["NAMA_KPH", "MEAN"]) as k:
    for nama, rata in k:
        print("%s: NDVI rata-rata %.3f" % (nama, rata))
arcpy.management.Delete(r"memory\sempadan") # ruang memory memakai RAM: bersihkan setelah selesai
arcpy.CheckInExtension("Spatial")
```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. Nama field `Join_Count` dan cara membaca band lewat `Band_n` ditulis dari dokumentasi Esri dan perlu diuji pada Pro Anda. Ruang kerja memory memakai RAM, jadi bersihkan dengan `arcpy.management.Delete`. [CEK]

BAGIAN C | ArcMap 10.8

Bagian C: Alat geoprocessing lewat ArcPy di ArcMap 10.8

Di ArcMap, nama alat memakai bentuk lama: `arcpy.Buffer_analysis`, `arcpy.Intersect_analysis`, dan `arcpy.SpatialJoin_analysis`. Ruang kerja semmentarnya bernama `in_memory`, bukan `memory`. Pembagian bilangan memakai `10000.0` karena Python 2.7.

PYTHON (ARCPY, PYTHON 2.7)

```
# -*- coding: utf-8 -*-
# [SKRIP 2.1-ArcMap: Rantai alat: sempadan sungai, titik panas, dan NDVI per KPH (Python 2.7)]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi Anda. Zonal Statistics as Table dan Float butuh ekstensi Spatial Analyst.
import arcpy
from arcpy.sa import Raster, Float, ZonalStatisticsAsTable

DATA = r"D:\Latihan\paket-i4"
HASIL = r"D:\Latihan\hasil_i4"
arcpy.env.workspace = DATA + r"\shp"
arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")
```

```
# Keluaran sementara di ArcMap: ruang kerja in_memory (di Pro namanya memory)
arcpy.Buffer_analysis("Sungai.shp", r"in_memory\sempadan", "30 Meters", dissolve_option="ALL")
arcpy.Intersect_analysis(["Batas_KPH.shp", r"in_memory\sempadan"], r"in_memory\irisan")
arcpy.SpatialJoin_analysis("Batas_KPH.shp", "Hotspot_Nasional.shp", r"in_memory\kph_titik",
                           "JOIN_ONE_TO_ONE", "KEEP_ALL", match_option="INTERSECT") # menambah Join_Count

sempadan = {}
with arcpy.da.SearchCursor(r"in_memory\irisan", ["NAMA_KPH", "SHAPE@AREA"]) as k:
    for nama, luas in k:
        sempadan[nama] = sempadan.get(nama, 0) + luas / 10000.0
with arcpy.da.SearchCursor(r"in_memory\kph_titik", ["NAMA_KPH", "Join_Count", "SHAPE@AREA"]) as k:
    for nama, jml, luas in k:
        print("%s: luas %.2f ha, sempadan %.2f ha, titik panas %d"
              % (nama, luas / 10000.0, sempadan.get(nama, 0), jml))

# NDVI = (NIR - Merah) / (NIR + Merah); band 4 = NIR, band 1 = Merah
nir = Float(Raster(DATA + r"\Citra_KPH.tif\Band_4"))
merah = Float(Raster(DATA + r"\Citra_KPH.tif\Band_1"))
ndvi = (nir - merah) / (nir + merah)
ndvi.save(HASIL + r"\NDVI_KPH.tif")
ZonalStatisticsAsTable("Batas_KPH.shp", "NAMA_KPH", ndvi, r"in_memory\ndvi_zonal", "DATA", "MEAN")
with arcpy.da.SearchCursor(r"in_memory\ndvi_zonal", ["NAMA_KPH", "MEAN"]) as k:
    for nama, rata in k:
        print("%s: NDVI rata-rata %.3f" % (nama, rata))
arcpy.Delete_management("in_memory") # bersihkan seluruh ruang in_memory
arcpy.CheckInExtension("Spatial")
```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. Ekstensi Spatial Analyst wajib aktif untuk Float, Raster, dan ZonalStatisticsAsTable. [CEK]

Cek paham

1. Alat vektor dijalankan dengan OUTPUT berisi "TEMPORARY_OUTPUT". Apa isi hasil["OUTPUT"]?
2. Layer raster bernama Citra_KPH. Bagaimana menulis band keempat dalam ekspresi raster calculator?
3. Mengapa DISTANCE: 30 pada layer berkoordinat derajat memberi hasil yang salah?

Jawaban:

1. Sebuah layer vektor, yang bisa langsung dipakai alat berikutnya.
2. "Citra_KPH@4", yaitu nama layer, tanda @, lalu nomor band.
3. Satuan jarak mengikuti CRS layer. Pada derajat, angka 30 berarti 30 derajat. Ubah ke CRS metrik (UTM) lebih dulu.

Kesalahan umum

- Lupa `["OUTPUT"]`. `processing.run` mengembalikan kamus, bukan layer. Ambil isinya dengan kunci "OUTPUT".
- Menganggap keluaran raster sementara sebagai layer. Isinya teks jalur. Bungkus dengan `QgsRasterLayer`.
- Nama layer di ekspresi raster tidak cocok. Pada uji, nama yang salah menghasilkan galat perhitungan. Samakan nama di ekspresi dengan nama layer.
- Hasil tidak muncul di peta. Layer sementara baru tampil setelah `addMapLayer`.

Ringkasan dan latihan

Ringkasan: `processing.run(ID, kamus)` menjalankan alat apa pun dan mengembalikan kamus hasil. `TEMPORARY_OUTPUT` cocok untuk hasil antara. Vektor kembali sebagai layer, raster sebagai jalur teks. Alat bisa dirantai tanpa berkas perantara.

Latihan:

1. Ubah lebar sempadan menjadi 50 m pada Skrip 2.2. Perhatikan perubahan luas sempadan tiap petak. (Total tiga petak seharusnya 6,55 ha.)
2. Tambahkan statistik 5 (minimum) dan 6 (maksimum) pada `STATISTICS` di Skrip 2.3. Cetak nilai NDVI terkecil dan terbesar per petak.

Tabel perbandingan: alat dari skrip

Hal	QGIS	ArcGIS Pro	ArcMap 10.8
Memanggil alat	<code>processing.run("native:buffer", {...})</code>	<code>arcpy.analysis.Buffer(...)</code>	<code>arcpy.Buffer_analysis(...)</code>
Hasil antara	<code>TEMPORARY_OUTPUT</code>	<code>memory\nama</code>	<code>in_memory\nama</code>
Hitung titik per poligon	<code>native:countpointsinpoligon</code>	<code>SpatialJoin + Join_Count [CEK]</code>	<code>SpatialJoin_Analysis + Join_Count [CEK]</code>
Raster calculator	<code>native:rastercalc</code>	<code>Aljabar peta di arcpy.sa [CEK]</code>	<code>Aljabar peta di arcpy.sa [CEK]</code>
Statistik zonal	<code>native:zonalstatisticsfb</code>	<code>ZonalStatisticsAsTable [CEK]</code>	<code>ZonalStatisticsAsTable [CEK]</code>
Ekstensi	Tidak perlu	<code>Spatial Analyst [CEK]</code>	<code>Spatial Analyst [CEK]</code>

BAB 3: Skrip Batch: Satu Resep untuk Banyak Berkas

Studi kasus: "Lima puluh ubin foto drone"

Tim drone menyerahkan puluhan ubin foto udara. Kepala Seksi minta NDVI untuk setiap ubin sebelum akhir pekan. Mengklik raster calculator satu per satu akan menghabiskan sore Anda. Lebih buruk lagi, QGIS sering membeku saat satu ubin diproses. Di buku ini dipakai tiga ubin agar mudah dicoba, tetapi skripnya sama untuk lima puluh.

Konsep: batch dalam tiga kalimat

Batch berarti menjalankan pekerjaan yang sama berulang kali pada banyak masukan. Anda menulis resepnya sekali, lalu perulangan `for` memasaknya untuk tiap berkas. Untuk pekerjaan berat, **QgsTask** menyerahkan resep itu ke "juru masak" di latar belakang, sementara Anda, sang "pelayan", tetap bisa melayani QGIS.

Istilah baru bab ini:

- **Batch**: pekerjaan sama, banyak masukan.
- **Pola berkas** (`glob`): cara mencari banyak berkas sekaligus, misalnya `*.tif`.
- **Latar belakang**: jalur kerja terpisah yang tidak membekukan layar QGIS.
- **QgsTask**: tugas latar belakang milik QGIS.
- **Fungsi selesai**: fungsi yang dipanggil setelah tugas berakhir.

Ilustrasi 3.1: Satu resep, banyak berkas

Satu resep, banyak berkas: perulangan (for)



Skema; bukan tangkapan layar

BAGIAN A | QGIS

Bagian A: Batch di QGIS

A.1 Tanpa menulis kode: batch bawaan

QGIS punya batch bawaan. Klik kanan sebuah alat di Processing Toolbox, lalu pilih **Execute as batch process**. Anda mengisi tabel: satu baris untuk satu pekerjaan. Cara ini cocok untuk beberapa berkas. Untuk puluhan berkas, mengisi tabel itu melelahkan, dan nama keluaran sulit diatur. Di sinilah skrip unggul.

A.2 Skrip batch sederhana

Skrip 3.1 mengumpulkan semua .tif di folder, menghitung NDVI tiap ubin, lalu mencetak rata-ratanya. Pada ubin ini band 1 adalah NIR dan band 2 adalah Merah.

PYTHON (PYQGIS) 3.1

```
# [SKRIP 3.1: Batch sederhana: NDVI untuk semua ubin ortofoto dalam satu folder]
# Penulis: Badar Mubarak Yogaswara
import glob
import os
import processing
from qgis.core import QgsRasterLayer

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
folder_masuk = DATA + "/Ortofoto_Mentah"
folder_keluar = HASIL + "/NDVI_ubin"
os.makedirs(folder_keluar, exist_ok=True)      # buat folder bila belum ada

# 1. Kumpulkan semua berkas .tif, diurutkan agar hasilnya selalu berurutan sama
daftar = sorted(glob.glob(folder_masuk + "/*.tif"))
print("Ditemukan", len(daftar), "ubin")

# 2. Ulangi pekerjaan yang sama untuk tiap berkas. Pada ubin ini band 1 = NIR dan band 2 = Merah.
berhasil = []
for jalur in daftar:
    nama = os.path.basename(jalur)                # contoh: ubin_01.tif
    keluar = folder_keluar + "/" + "NDVI_" + nama
    ubin = QgsRasterLayer(jalur, "ubin", "gdal")    # nama "ubin" dipakai di ekspresi
    processing.run("native:rastercalc", {
        "EXPRESSION": '("ubin@1" - "ubin@2") / ("ubin@1" + "ubin@2")',
        "LAYERS": [ubin], "OUTPUT": keluar})
    hasil = QgsRasterLayer(keluar, "hasil", "gdal")
    rata = processing.run("native:rasterlayerstatistics", {"INPUT": hasil, "BAND": 1})["MEAN"]
    print(" %s -> %s | NDVI rata-rata %.3f" % (nama, os.path.basename(keluar), rata))
    berhasil.append(keluar)

print("Selesai:", len(berhasil), "dari", len(daftar), "berkas")
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 3.1

```
Ditemukan 3 ubin
ubin_01.tif -> NDVI_ubin_01.tif | NDVI rata-rata 0.471
ubin_02.tif -> NDVI_ubin_02.tif | NDVI rata-rata 0.471
ubin_03.tif -> NDVI_ubin_03.tif | NDVI rata-rata 0.467
Selesai: 3 dari 3 berkas
```

Bedah kode: `glob.glob(folder + "/*.tif")` mengembalikan semua jalur berkas berakhiran `.tif`. `sorted(...)` menjaga urutannya tetap sama setiap kali dijalankan. `os.path.basename(jalur)` mengambil nama berkas saja, misalnya `ubin_01.tif`. Nama itu dipakai untuk menamai hasil. `os.makedirs(..., exist_ok=True)` membuat folder keluaran bila belum ada, dan tidak protes bila sudah ada. Saat statistik dihitung, QGIS dapat membuat berkas pendamping `.aux.xml` di samping raster. Berkas itu aman dibiarkan.

A.3 Satu putaran untuk tiap fitur

Perulangan tidak harus atas berkas. Skrip 3.2 berputar atas **fitur**: untuk tiap KPH, skrip memilih satu poligon lalu memotong CHM dengannya. Hasilnya satu raster per KPH.

Ada satu trik penting. Nama KPH disisipkan ke ekspresi pemilih dengan `%s`. Tanda kutip tunggal di sekitar nama ditulis `\'`, supaya Python tidak menganggapnya penutup teks.

PYTHON (PYQGIS) 3.2

```
# [SKRIP 3.2: Batch per fitur: potong raster tinggi tajuk (CHM) untuk tiap KPH]
# Penulis: Badar Mubarak Yogaswara

import os
import processing
from qgis.core import QgsVectorLayer, QgsRasterLayer

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
folder_keluar = HASIL + "/CHM_per_KPH"
os.makedirs(folder_keluar, exist_ok=True)

batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
dsm = QgsRasterLayer(DATA + "/DSM_Drone.tif", "DSM", "gdal")
dtm = QgsRasterLayer(DATA + "/DTM_Drone.tif", "DTM", "gdal")

# 1. CHM untuk seluruh area (satu kali saja, di luar pengulangan)
chm_jalur = processing.run("native:rastercalc", {
    "EXPRESSION": "DSM@1" - "DTM@1", "LAYERS": [dsm, dtm], "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]

# 2. Satu putaran untuk tiap KPH: pilih satu poligon, lalu potong CHM dengannya
for fitur in batas.getFeatures():
    nama = fitur["NAMA_KPH"] # contoh: KPH Alpha
    aman = nama.replace(" ", "_") # untuk nama berkas: KPH_Alpha
    satu_kph = processing.run("native:extractbyexpression", {
        "INPUT": batas, "EXPRESSION": '"NAMA_KPH" = \'' + nama + '\'',
        "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
    keluar = folder_keluar + "/CHM_" + aman + ".tif"
    processing.run("gdal:cliprasterbymasklayer", {
        "INPUT": chm_jalur, "MASK": satu_kph, "CROP_TO_OUTLINE": True,
        "NODATA": -9999, "OUTPUT": keluar})
    hasil = QgsRasterLayer(keluar, aman, "gdal")
    st = processing.run("native:rasterlayerstatistics", {"INPUT": hasil, "BAND": 1})
    print("%-10s %4d x %4d piksel | rata-rata %.2f m | tertinggi %.2f m" % (
        aman, hasil.width(), hasil.height(), st["MEAN"], st["MAX"]))
    print("Berkas di folder:", sorted(os.listdir(folder_keluar)))
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 3.2

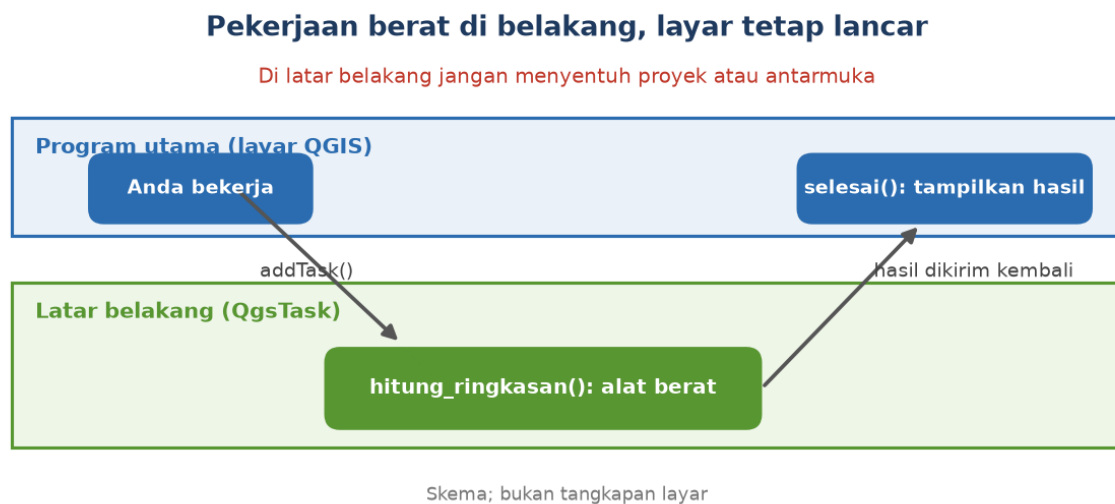
KPH_Alpha 400 x 400 piksel | rata-rata 2.60 m | tertinggi 13.94 m
 KPH_Beta 400 x 400 piksel | rata-rata 2.69 m | tertinggi 13.96 m
 KPH_Gamma 800 x 400 piksel | rata-rata 2.56 m | tertinggi 13.96 m
 Berkas di folder: ['CHM_KPH_Alpha.tif', 'CHM_KPH_Alpha.tif.aux.xml', 'CHM_KPH_Beta.tif',
 'CHM_KPH_Beta.tif.aux.xml', 'CHM_KPH_Gamma.tif']

Ukuran hasil bisa dicek dengan akal sehat. Alpha dan Beta berukuran 200 m x 200 m dengan piksel 0,5 m, yaitu 400 x 400 piksel. Gamma dua kali lebih lebar, yaitu 800 x 400 piksel.

A.4 Batch di latar belakang dengan QgsTask

Perulangan di Skrip 3.1 membuat layar QGIS "menunggu" sampai semua ubin selesai. Pada lima puluh ubin, itu terasa seperti macet. Skrip 3.3 memindahkan pekerjaan yang sama ke latar belakang.

Ilustrasi 3.2: Program utama dan latar belakang



Susunannya selalu dua fungsi:

1. **Fungsi tugas** (`hitung_ndvi`) berjalan di latar belakang. Argumen pertamanya selalu `task`. Fungsi ini boleh menjalankan alat dan membaca berkas. Fungsi ini **tidak boleh** menyentuh proyek atau antarmuka QGIS.
2. **Fungsi selesai** (`selesai`) berjalan di program utama setelah tugas berakhir. Di sinilah hasil ditambahkan ke proyek atau ditampilkan.

PYTHON (PYQGIS) 3.3

```
# [SKRIP 3.3: Batch di latar belakang dengan QgsTask supaya QGIS tidak macet]
# Penulis: Badar Mubarak Yogaswara
import glob
import os
import processing
from qgis.core import QgsApplication, QgsTask, QgsMessageLog, Qgis, QgsRasterLayer, QgsProject

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
folder_masuk = DATA + "/Ortofoto_Mentah"
folder_keluar = HASIL + "/NDVI_latar"
os.makedirs(folder_keluar, exist_ok=True)
```

```
def hitung_ndvi(task, folder_in, folder_out):
    """Berjalan di LATAR BELAKANG. Jangan menyentuh proyek atau antarmuka di sini."""
    daftar = sorted(glob.glob(folder_in + "/*.tif"))
    keluar = []
    for i, jalur in enumerate(daftar):
        if task.isCanceled():
            # pengguna menekan Cancel
            return None
        tujuan = folder_out + "/NDVI_" + os.path.basename(jalur)
        ubin = QgsRasterLayer(jalur, "ubin", "gdal")
        processing.run("native:rastercalc", {
            "EXPRESSION": '("ubin@1" - "ubin@2") / ("ubin@1" + "ubin@2")',
            "LAYERS": [ubin], "OUTPUT": tujuan})
        keluar.append(tujuan)
        task.setProgress(100 * (i + 1) / len(daftar)) # bilah kemajuan di pojok kanan bawah QGIS
    return keluar

def selesai(exception, hasil=None):
    """Berjalan di PROGRAM UTAMA setelah tugas berakhir. Aman untuk menyentuh proyek."""
    if exception is not None:
        QgsMessageLog.logMessage("Tugas gagal: %s" % exception, "Batch NDVI", Qgs.Critical)
    elif hasil is None:
        QgsMessageLog.logMessage("Tugas dibatalkan", "Batch NDVI", Qgs.Warning)
    else:
        for jalur in hasil:
            nama = os.path.basename(jalur)[-4:]
            QgsProject.instance().addMapLayer(QgsRasterLayer(jalur, nama, "gdal"))
            QgsMessageLog.logMessage("Selesai: %d berkas" % len(hasil), "Batch NDVI", Qgs.Success)
            print("Tugas selesai:", len(hasil), "berkas NDVI ditambahkan ke proyek")

tugas = QgsTask.fromFunction("Hitung NDVI semua ubin", hitung_ndvi, on_finished=selesai,
                             folder_in=folder_masuk, folder_out=folder_keluar)
QgsApplication.taskManager().addTask(tugas)
print("Tugas dikirim ke latar belakang. QGIS tetap bisa dipakai.")
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 3.3

Tugas dikirim ke latar belakang. QGIS tetap bisa dipakai.
Tugas selesai: 3 berkas NDVI ditambahkan ke proyek

Bedah kode: `QgsTask.fromFunction("Judul", fungsi, on_finished=..., nama=nilai)` membuat tugas. Argumen bernama (seperti `folder_in`) diteruskan ke fungsi tugas. `task.setProgress(angka)` mengisi bilah kemajuan dengan nilai 0 sampai 100. `task.isCanceled()` memeriksa apakah pengguna menekan Cancel. Nilai yang di-return dari fungsi tugas diterima fungsi selesai sebagai argumen kedua. Bila fungsi tugas return `None`, skrip ini menganggapnya dibatalkan. `QgsApplication.taskManager().addTask(tugas)` mengirim tugas. Simpan tugas dalam variabel (di sini `tugas`) agar tidak hilang. Kemajuan tugas terlihat di bilah status QGIS, dan pesan di panel **Log Messages**. [CEK: tampilan visualnya]

Batas latar belakang: Hanya jalankan alat yang aman di latar belakang. Alat bawaan seperti `native:rastercalc` aman, dan sudah diuji di sini. Menambah layer ke proyek atau menyentuh `iface` dari fungsi tugas dapat membuat QGIS macet atau tertutup sendiri. Lakukan itu di fungsi selesai.

BAGIAN B | ArcGIS Pro

Bagian B: Batch di ArcGIS Pro

Pro tidak punya padanan langsung `QgsTask`. Pekerjaan banyak berkas ditulis sebagai skrip batch biasa. Alat `arcpy.ListRasters` mendaftarkan raster di `arcpy.env.workspace`. Perulangan `for` memprosesnya satu per satu. Untuk pekerjaan panjang, kemajuan bisa ditampilkan lewat `arcpy.SetProgressor` bila skrip dijadikan alat skrip. [CEK] Esri juga menyediakan opsi paralel (`Parallel Processing Factor`) yang perlu Anda coba sendiri. [CEK]

PYTHON (ARCPY, PYTHON 3)

```
# [SKRIP 3.1-Pro: Batch: NDVI semua ubin dan potongan CHM per KPH dengan ArcPy]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi Anda. Butuh ekstensi Spatial Analyst.
import os
import arcpy
from arcpy.sa import Raster, Float, ExtractByMask

DATA = r"D:\Latihan\paket-i4"
HASIL = r"D:\Latihan\hasil_i4"
arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")
os.makedirs(HASIL + r"\NDVI_ubin", exist_ok=True)

# A. Semua ubin dalam satu folder (band 1 = NIR, band 2 = Merah)
arcpy.env.workspace = DATA + r"\Ortofoto_Mentah"
daftar = arcpy.ListRasters("*.TIF")
for i, nama in enumerate(daftar, start=1):
    b1 = Float(Raster(os.path.join(DATA, "Ortofoto_Mentah", nama, "Band_1")))
    b2 = Float(Raster(os.path.join(DATA, "Ortofoto_Mentah", nama, "Band_2")))
    ((b1 - b2) / (b1 + b2)).save(os.path.join(HASIL, "NDVI_ubin", "NDVI_" + nama))
    print("%d/%d selesai: %s" % (i, len(daftar), nama))

# B. Satu putaran per KPH: pilih satu poligon, potong CHM dengannya
chm = Raster(DATA + r"\DSM_Drone.tif") - Raster(DATA + r"\DTM_Drone.tif")
batas = DATA + r"\shp\Batas_KPH.shp"
with arcpy.da.SearchCursor(batas, ["NAMA_KPH"]) as k:
    nama_kph = [b[0] for b in k]
for nama in nama_kph:
    arcpy.analysis.Select(batas, r"memory\satu_kph", "NAMA_KPH = '%s'" % nama)
    keluar = os.path.join(HASIL, "CHM_" + nama.replace(" ", "_") + ".tif")
    ExtractByMask(chm, r"memory\satu_kph").save(keluar)
    print("CHM tersimpan untuk", nama)
arcpy.CheckInExtension("Spatial")
```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. Urutan band mengikuti Skrip 3.1: pastikan `Band_1` adalah NIR dan `Band_2` adalah Merah pada ortofoto Anda. [CEK]

BAGIAN C | ArcMap 10.8

Bagian C: Batch di ArcMap 10.8

ArcMap memakai Python 2.7, jadi `os.makedirs` tidak punya `exist_ok` dan harus diperiksa lebih dulu dengan `os.path.exists`. ArcMap tidak punya padanan `QgsTask`. Esri menyediakan pengaturan pemrosesan latar belakang di opsi geoprocessing, yang berlaku untuk alat yang dijalankan dari antarmuka, bukan untuk skrip mandiri. [CEK]

PYTHON (ARCPY, PYTHON 2.7)

```
# -*- coding: utf-8 -*-
# [SKRIP 3.1-ArcMap: Batch: NDVI semua ubin dan potongan CHM per KPH (Python 2.7)]
# Penulis: Badar Mubarok Yogaswara
# [CEK] Cocokkan dengan versi Anda. Butuh ekstensi Spatial Analyst.
import os
import arcpy
from arcpy.sa import Raster, Float, ExtractByMask

DATA = r"D:\Latihan\paket-i4"
HASIL = r"D:\Latihan\hasil_i4"
arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")
if not os.path.exists(HASIL + r"\NDVI_ubin"):          # Python 2.7 tidak punya exist_ok
    os.makedirs(HASIL + r"\NDVI_ubin")

# A. Semua ubin dalam satu folder (band 1 = NIR, band 2 = Merah)
arcpy.env.workspace = DATA + r"\Ortofoto_Mentah"
daftar = arcpy.ListRasters("*.TIF")
for i, nama in enumerate(daftar, 1):
    b1 = Float(Raster(os.path.join(DATA, "Ortofoto_Mentah", nama, "Band_1")))
    b2 = Float(Raster(os.path.join(DATA, "Ortofoto_Mentah", nama, "Band_2")))
    ((b1 - b2) / (b1 + b2)).save(os.path.join(HASIL, "NDVI_ubin", "NDVI_" + nama))
    print("%d/%d selesai: %s" % (i, len(daftar), nama))

# B. Satu putaran per KPH: pilih satu poligon, potong CHM dengannya
chm = Raster(DATA + r"\DSM_Drone.tif") - Raster(DATA + r"\DTM_Drone.tif")
batas = DATA + r"\shp\Batas_KPH.shp"
with arcpy.da.SearchCursor(batas, ["NAMA_KPH"]) as k:
    nama_kph = [b[0] for b in k]
for nama in nama_kph:
    arcpy.Select_analysis(batas, r"in_memory\satu_kph", "\"NAMA_KPH\" = '%s'" % nama)
    keluar = os.path.join(HASIL, "CHM_" + nama.replace(" ", "_") + ".tif")
    ExtractByMask(chm, r"in_memory\satu_kph").save(keluar)
    print("CHM tersimpan untuk %s" % nama)
arcpy.CheckInExtension("Spatial")
```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. Ekstensi Spatial Analyst wajib aktif. [CEK]

Cek paham

1. Mengapa daftar berkas dibungkus `sorted(...)`?
2. Di fungsi mana layer hasil boleh ditambahkan ke proyek pada skrip `QgsTask`?
3. Apa gunanya `task.isCanceled()`?

Jawaban:

1. Agar urutan pemrosesan dan hasilnya selalu sama setiap kali dijalankan.
2. Di fungsi selesai, yang berjalan di program utama. Fungsi tugas berjalan di latar belakang dan tidak boleh menyentuh proyek.
3. Untuk memeriksa apakah pengguna menekan Cancel. Bila ya, fungsi tugas berhenti dengan rapi.

Kesalahan umum

- **Mengira urutan band.** Rumus NDVI memakai band NIR dan Merah. Pada ubin ini band 1 = NIR dan band 2 = Merah. Periksa dulu pada data Anda.
- **Satu berkas rusak menghentikan seluruh batch.** Ganti folder pada Skrip 3.1 menjadi `Ortofoto_Campur`. Tiga ubin pertama selesai, lalu ubin rusak menghentikan skrip dengan pesan `Error creating output file`. Pesan itu tidak menyebut berkas rusak. Bab 4 menjawab masalah ini.
- **Menyentuh proyek dari latar belakang.** Pindahkan ke fungsi selesai.
- **Folder keluaran belum ada.** Buat dengan `os.makedirs`.

Ringkasan dan latihan

Ringkasan: perulangan `for` menjalankan resep yang sama untuk banyak berkas atau fitur. `glob` mencari berkas, dan `sorted` menertibkan urutan. `QgsTask` memindahkan pekerjaan berat ke latar belakang: fungsi tugas bekerja, fungsi selesai menampilkan hasil.

Latihan:

1. Ubah Skrip 3.1 agar nama hasil berbunyi `ndvi_01.tif`, `ndvi_02.tif`, dan seterusnya (petunjuk: `enumerate`).
2. Ubah Skrip 3.3 agar fungsi selesai mencetak rata-rata NDVI tiap hasil. Gunakan `native:rasterlayerstatistics`.

Tabel perbandingan: batch

Hal	QGIS	ArcGIS Pro	ArcMap 10.8
Batch tanpa kode	Execute as batch process	Batch pada dialog alat [CEK]	Batch pada menu klik kanan alat [CEK]
Daftar raster	<code>glob.glob("*.tif")</code>	<code>arcpy.ListRasters</code>	<code>arcpy.ListRasters</code>
Perulangan atas fitur	<code>getFeatures()</code>	<code>arcpy.da.SearchCursor</code>	<code>arcpy.da.SearchCursor</code>
Tugas latar belakang	<code>QgsTask</code> (diuji)	Tidak ada padanan langsung; skrip batch dan alat skrip [CEK]	Tidak ada padanan langsung; opsi latar belakang [CEK]
Bilah kemajuan	<code>task.setProgress</code>	<code>arcpy.SetProgressor</code> [CEK]	Tidak ada padanan skrip [CEK]

BAB 4: Penanganan Galat: Skrip yang Tidak Mudah Menyerah

Studi kasus: "Kiriman dari enam tim"

Enam tim lapangan mengirim data titik survei. Malam hari, skrip penggabung Anda berhenti di tengah jalan dengan pesan panjang berbahasa Inggris. Anda tidak tahu berkas mana yang bermasalah, dan berkas sebelumnya sudah terlanjur diproses setengah. Keesokan harinya Kepala Seksi hanya bertanya: berapa kiriman yang diterima, mana yang ditolak, dan mengapa. Skrip yang baik menjawab semua itu sendiri.

Konsep: penanganan galat dalam tiga kalimat

Galat (exception) adalah tanda dari Python bahwa ada yang salah, dan tanpa penanganan skrip langsung berhenti. Dengan **try-except**, Anda menangkap galat itu, mencatat sebabnya, lalu meneruskan pekerjaan. Ini seperti petugas gudang yang memeriksa tiap kiriman: barang rusak dikembalikan dengan catatan, bukan seluruh gudang ditutup.

Istilah baru bab ini:

- **Galat** (exception): sinyal bahwa ada yang gagal.
- **try-except**: blok yang mencoba sesuatu dan menangkap galatnya.
- **Validasi**: pemeriksaan masukan sebelum dipakai.
- **Log**: catatan hasil tiap langkah, yang bisa dibaca belakangan.
- **QgsProcessingException**: galat khusus yang dikeluarkan alat Processing.

Ilustrasi 4.1: Tanpa dan dengan penanganan galat



BAGIAN A | QGIS

Bagian A: Penanganan galat di QGIS

A.1 Memeriksa sebelum memakai

Skrip 4.1 mendefinisikan fungsi `muat_vektor` yang **menolak** masukan bermasalah dengan pesan jelas: berkas tidak ada, tidak bisa dibuka, field wajib hilang, atau layer kosong. Fungsi itu lalu dicoba pada empat berkas.

PYTHON (PYQGIS) 4.1

```
# [SKRIP 4.1: Memeriksa masukan sebelum dipakai dan menangkap galat dengan try-except]
# Penulis: Badar Mubarak Yogaswara
import os
from qgis.core import QgsVectorLayer

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

def muat_vektor(jalur, nama, field_wajib=()):
    """Membuka layer vektor. Bila ada yang tidak beres, berhenti dengan pesan yang jelas."""
    if not os.path.exists(jalur):
        raise FileNotFoundError("berkas tidak ada: " + jalur)
    layer = QgsVectorLayer(jalur, nama, "ogr")
    if not layer.isValid():
        raise ValueError("berkas tidak bisa dibuka (mungkin rusak): " + os.path.basename(jalur))
    kurang = [f for f in field_wajib if layer.fields().indexOf(f) < 0]
    if kurang:
        raise ValueError("field wajib tidak ada: " + ", ".join(kurang))
    if layer.featureCount() == 0:
        raise ValueError("layer kosong, tidak ada fitur")
    return layer

# Empat berkas dicoba satu per satu. Satu gagal tidak boleh menghentikan yang lain.
kasus = [
    DATA + "/Kiriman_Lapangan/tim_a.gpkg",      # baik
    DATA + "/Kiriman_Lapangan/tim_x.gpkg",      # tidak ada
    DATA + "/Kiriman_Lapangan/tim_e.gpkg",      # rusak
    DATA + "/Kiriman_Lapangan/tim_f.gpkg",      # field Kondisi tidak ada
]
for jalur in kasus:
    nama = os.path.basename(jalur)
    try:
        layer = muat_vektor(jalur, nama, field_wajib=["Kondisi", "Tinggi_Phn"])
    except FileNotFoundError as e:
        print("[LEWATI] %s -> %s" % (nama, e))
    except ValueError as e:
        print("[LEWATI] %s -> %s" % (nama, e))
    else:
        # blok else: hanya jalan bila TIDAK ada galat
        print("[OK] %s -> %d fitur" % (nama, layer.featureCount()))
    finally:
        # blok finally: selalu jalan, berhasil atau gagal
        print("selesai memeriksa", nama)

# Galat hitungan yang sering terjadi: pembagian dengan nol
def persen(bagian, total):
    try:
        return 100 * bagian / total
    except ZeroDivisionError:
        return None          # tidak ada jawaban yang masuk akal
```

```
print("Persen 3 dari 12:", persen(3, 12))
print("Persen 3 dari 0:", persen(3, 0))
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 4.1

```
[OK] tim_a.gpkg -> 6 fitur
    selesai memeriksa tim_a.gpkg
[LEWATI] tim_x.gpkg -> berkas tidak ada: D:/Latihan/paket-i4/Kiriman_Lapangan/tim_x.gpkg
    selesai memeriksa tim_x.gpkg
[LEWATI] tim_e.gpkg -> berkas tidak bisa dibuka (mungkin rusak): tim_e.gpkg
    selesai memeriksa tim_e.gpkg
[LEWATI] tim_f.gpkg -> field wajib tidak ada: Kondisi
    selesai memeriksa tim_f.gpkg
Persen 3 dari 12: 25.0
Persen 3 dari 0: None
```

Bedah kode: `raise ValueError("...")` sengaja memicu galat dengan pesan buatan Anda. Pesan yang jelas menghemat waktu mencari sebab. `try`: memuat perintah yang mungkin gagal. Tiap `except JenisGalat as e`: menangkap satu jenis galat. `e` berisi pesannya. `else`: hanya jalan bila **tidak** ada galat. `finally`: selalu jalan, baik berhasil maupun gagal. Tangkap jenis galat yang Anda perkirakan. Hindari `except`: kosong yang diam-diam menelan semua galat, karena kesalahan Anda sendiri ikut tersembunyi.

A.2 Batch tahan galat dengan laporan

Skrip 4.2 memproses semua kiriman. Untuk tiap berkas, skrip memeriksa tiga hal berlapis: bisa dibuka, field lengkap, lalu CRS sesuai. Bila CRS berbeda tetapi masih bisa diperbaiki, skrip mengubahnya ke EPSG:32749 dan mencatatnya. Berkas yang tidak bisa diperbaiki ditolak beserta sebabnya. Setelah digabung, skrip memeriksa titik yang jatuh di luar semua KPH.

Ilustrasi 4.2: Pemeriksaan berlapis

Pemeriksaan berlapis untuk satu kiriman lapangan



Skema; bukan tangkapan layar

PYTHON (PYQGIS) 4.2

```
# [SKRIP 4.2: Batch tahan galat: menggabungkan kiriman lapangan, mencatat yang gagal]
```

```

# Penulis: Badar Mubarak Yogaswara
import csv
import glob
import os
import processing
from qgis.core import QgsProject, QgsVectorLayer, QgsCoordinateReferenceSystem

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
os.makedirs(HASIL, exist_ok=True)

CRS_TUJUAN = "EPSG:32749"
FIELD_WAJIB = ["Kondisi", "Tinggi_Phn", "Foto"]
log = [] # satu baris catatan untuk tiap berkas: (berkas, status, keterangan)
bagus = [] # layer yang lolos pemeriksaan

for jalur in sorted(glob.glob(DATA + "/Kiriman_Lapangan/*.gpkg")):
    nama = os.path.basename(jalur)
    try:
        layer = QgsVectorLayer(jalur, nama, "ogr")
        if not layer.isValid():
            raise ValueError("berkas rusak atau bukan GeoPackage")
        kurang = [f for f in FIELD_WAJIB if layer.fields().indexOf(f) < 0]
        if kurang:
            raise ValueError("field hilang: " + ", ".join(kurang))
        catatan = "sesuai"
        if layer.crs().authid() != CRS_TUJUAN:
            asal = layer.crs().authid()
            layer = processing.run("native:reprojectlayer", {
                "INPUT": layer, "TARGET_CRS": QgsCoordinateReferenceSystem(CRS_TUJUAN),
                "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
            catatan = "CRS %s diubah ke %s" % (asal, CRS_TUJUAN)
        bagus.append(layer)
        log.append((nama, "DITERIMA", "%d titik; %s" % (layer.featureCount(), catatan)))
    except Exception as e: # satu berkas gagal, yang lain tetap diproses
        log.append((nama, "DITOLAK", str(e)))

# Gabungkan yang lolos
gabungan = processing.run("native:mergevectorlayers", {
    "LAYERS": bagus, "CRS": QgsCoordinateReferenceSystem(CRS_TUJUAN),
    "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
gabungan.setName("Titik_Survei_Gabungan")
QgsProject.instance().addMapLayer(gabungan)

# Pemeriksaan keberadaan: titik yang tidak jatuh di dalam KPH mana pun patut dicurigai
batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
cek = processing.run("native:joinattributesbylocation", {
    "INPUT": gabungan, "JOIN": batas, "JOIN_FIELDS": ["NAMA_KPH"], "METHOD": 1,
    "PREDICATE": [0], "DISCARD_NONMATCHING": False, "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
luar = [f["Foto"] for f in cek.getFeatures() if not f["NAMA_KPH"]]
for foto in luar:
    log.append((foto, "CURIGA", "titik jatuh di luar semua KPH"))

# Laporan: layer dan berkas CSV
with open(HASIL + "/log_kiriman.csv", "w", newline="", encoding="utf-8") as f:
    w = csv.writer(f)
    w.writerow(["berkas", "status", "keterangan"])
    w.writerows(log)
for baris in log:
    print("%-22s %-9s %s" % baris)
print("Ringkasan: %d diterima, %d ditolak, %d curiga; total titik gabungan %d" % (
    sum(1 for b in log if b[1] == "DITERIMA"), sum(1 for b in log if b[1] == "DITOLAK"),
    len(luar), gabungan.featureCount()))

```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 4.2

tim_a.gpkg	DITERIMA	6 titik; sesuai
tim_b.gpkg	DITERIMA	5 titik; sesuai
tim_c.gpkg	DITERIMA	4 titik; sesuai
tim_d.gpkg	DITERIMA	3 titik; CRS EPSG:4326 diubah ke EPSG:32749
tim_e.gpkg	DITOLAK	berkas rusak atau bukan GeoPackage
tim_f.gpkg	DITOLAK	field hilang: Kondisi
b_05.jpg	CURIGA	titik jatuh di luar semua KPH
Ringkasan: 4 diterima, 2 ditolak, 1 curiga; total titik gabungan 18		

Angka dapat dicek dari pembuat data: tim A mengirim 6 titik, tim B 5, tim C 4, dan tim D 3 (dikirim dalam derajat). Jumlahnya 18. Tim E mengirim berkas rusak dan tim F tidak punya field `Kondisi`, jadi keduanya ditolak. Satu titik tim B sengaja diberi koordinat salah ketik, sehingga ditandai curiga.

Perbaiki dengan jejak: Mengubah CRS otomatis itu praktis, tetapi selalu catat. Tanpa catatan, orang lain tidak tahu data itu pernah diubah. Titik curiga **tidak** dibuang oleh skrip. Keputusan ada pada Anda, bukan pada skrip.

A.3 Menangkap galat dari alat Processing

Alat Processing mengeluarkan `QgsProcessingException`. Skrip 4.3 membungkus `processing.run` dalam fungsi `jalankan_aman`. Fungsi itu mengembalikan pasangan (hasil, pesan), sehingga skrip pemanggil tidak perlu menulis `try` berulang kali.

PYTHON (PYQGIS) 4.3

```
# [SKRIP 4.3: Menangkap galat dari alat Processing dengan pembungkus kecil]
# Penulis: Badar Mubarak Yogaswara
import traceback
import processing
from qgis.core import QgsVectorLayer, QgsProcessingException, QgsMessageLog, Qgis

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

def jalankan_aman(alat, parameter):
    """Mengembalikan (hasil, pesan). Bila alat gagal: hasil = None dan pesan berisi sebabnya."""
    try:
        return processing.run(alat, parameter), ""
    except QgsProcessingException as e:
        # galat milik Processing
        QgsMessageLog.logMessage("%s gagal: %s" % (alat, e), "Otomasi KPH", Qgis.Warning)
        return None, str(e).replace("\n", " | ")
    except Exception:
        # galat Python lain: simpan jejak untuk diagnosis
        return None, "galat tak terduga: " + traceback.format_exc().splitlines()[-1]

sungai = QgsVectorLayer(DATA + "/Sungai.gpkg", "Sungai", "ogr")

# Kasus 1: berjalan normal
hasil, pesan = jalankan_aman("native:buffer", {
    "INPUT": sungai, "DISTANCE": 30, "OUTPUT": "TEMPORARY_OUTPUT"})
print("Kasus 1:", "OK, %d fitur" % hasil["OUTPUT"].featureCount() if hasil else "GAGAL: " + pesan)

# Kasus 2: masukan tidak ada
hasil, pesan = jalankan_aman("native:buffer", {
    "INPUT": DATA + "/tidak_ada.gpkg", "DISTANCE": 30, "OUTPUT": "TEMPORARY_OUTPUT"})
print("Kasus 2:", "OK" if hasil else "GAGAL: " + pesan)
```

```
# Kasus 3: nama alat salah ketik
hasil, pesan = jalankan_aman("native:buffr", {})
print("Kasus 3:", "OK" if hasil else "GAGAL: " + pesan)

# Kasus 4: parameter wajib (OVERLAY) tidak diisi
hasil, pesan = jalankan_aman("native:clip", {"INPUT": sungai, "OUTPUT": "TEMPORARY_OUTPUT"})
print("Kasus 4:", "OK" if hasil else "GAGAL: " + pesan)
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 4.3

```
Kasus 1: OK, 2 fitur
Kasus 2: GAGAL: Unable to execute algorithm | Could not load source layer for INPUT:
D:/Latihan/paket-i4/tidak_ada.gpkg not found
Kasus 3: GAGAL: Error: Algorithm native:buffr not found |
Kasus 4: GAGAL: Unable to execute algorithm | Could not load source layer for OVERLAY: no value specified for
parameter
```

Pesan galat QGIS berbahasa Inggris. Biasakan membaca baris terakhirnya, karena di situlah sebab yang sebenarnya, misalnya `Could not load source layer for INPUT`. Pada uji, berkas yang tidak ada menghasilkan pesan itu.

Gejala	Penyebab umum	Tindakan
<code>isValid()</code> bernilai False	Berkas hilang, rusak, atau formatnya salah	Tolak dan catat nama berkas
<code>IndexError</code> pada [0]	Nama layer tidak ketemu	Periksa ejaan, atau tangani daftar kosong
<code>Could not load source layer</code>	Masukan alat tidak terbaca	Periksa jalur dan <code>isValid()</code> sebelum memanggil alat
<code>Algorithm ... not found</code>	ID alat salah ketik	Cari lagi ID dengan Skrip 2.1
Hasil kosong tanpa galat	Filter atau CRS tidak cocok	Periksa jumlah fitur setiap langkah

BAGIAN B | ArcGIS Pro

Bagian B: Penanganan galat di ArcGIS Pro

Alat geoprocessing yang gagal memicu `arcpy.ExecuteError`. Pesan alat yang gagal dibaca dengan `arcpy.GetMessages(2)`. Menurut dokumentasi Esri, galat dibagi dua kelompok: galat geoprocessing (`arcpy.ExecuteError`) dan galat Python lain. Keduanya ditangkap terpisah. Dalam alat skrip, pesan dikirim ke jendela alat dengan `arcpy.AddError`.

Pemeriksaan masukan memakai `arcpy.Exists`, `arcpy.ListFields`, dan `arcpy.Describe(...).spatialReference.factoryCode`. Bila CRS berbeda, `arcpy.management.Project` mengubahnya. [CEK]

PYTHON (ARCPY, PYTHON 3)

```
# [SKRIP 4.1-Pro: Penanganan galat dengan ArcPy: periksa masukan lalu tangkap ExecuteError]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi Anda. Kiriman lapangan berbentuk Shapefile di folder Kiriman_Lapangan\shp.
import glob
import os
import sys
import arcpy

DATA = r"D:\Latihan\paket-i4"
FIELD_WAJIB = ["Kondisi", "Tinggi_Phn", "Foto"]
log = []

def periksa(jalur):
    """Berhenti dengan pesan jelas bila masukan tidak layak. Mengembalikan kode EPSG data."""
    if not arcpy.Exists(jalur):
        raise ValueError("data tidak ada atau tidak bisa dibuka")
    nama_field = [f.name for f in arcpy.ListFields(jalur)]
    kurang = [f for f in FIELD_WAJIB if f not in nama_field]
    if kurang:
        raise ValueError("field hilang: " + ", ".join(kurang))
    return arcpy.Describe(jalur).spatialReference.factoryCode

for jalur in sorted(glob.glob(DATA + r"\Kiriman_Lapangan\shp\*.shp")):
    nama = os.path.basename(jalur)
    try:
        epsg = periksa(jalur)
        if epsg != 32749:
            arcpy.management.Project(jalur, "memory\\" + nama[:-4], arcpy.SpatialReference(32749))
            log.append((nama, "DITERIMA", "CRS %s diubah ke 32749" % epsg))
        else:
            log.append((nama, "DITERIMA", "sesuai"))
    except ValueError as e:
        # galat milik skrip kita
        log.append((nama, "DITOLAK", str(e)))
    except arcpy.ExecuteError:
        # galat dari alat geoprocessing
        log.append((nama, "GAGAL ALAT", arcpy.GetMessages(2)))
    except Exception:
        # galat Python lain
        log.append((nama, "TAK TERDUGA", str(sys.exc_info()[1])))
for baris in log:
    print("%-14s %-12s %s" % baris)
```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. Skrip membaca Shapefile di Kiriman_Lapangan\shp, yaitu salinan kiriman yang disiapkan di paket data. [CEK]

BAGIAN C | ArcMap 10.8

Bagian C: Penanganan galat di ArcMap 10.8

Konsepnya sama: `arcpy.ExecuteError` dan `arcpy.GetMessages(2)`. Perbedaannya hanya nama alat (`arcpy.Project_management`) dan ruang sementara (`in_memory`). Bentuk `except ValueError as e` sudah berlaku di Python 2.7.

PYTHON (ARCPY, PYTHON 2.7)

```
# -*- coding: utf-8 -*-
# [SKRIP 4.1-ArcMap: Penanganan galat dengan ArcPy (Python 2.7)]
# Penulis: Badar Mubarak Yogaswara
```

```
# [CEK] Cocokkan dengan versi Anda. Kiriman lapangan diandaikan berbentuk Shapefile di folder
Kiriman_Lapangan\shp.
import glob
import os
import sys
import arcpy

DATA = r"D:\Latihan\paket-i4"
FIELD_WAJIB = ["Kondisi", "Tinggi_Ph", "Foto"]
log = []

def periksa(jalur):
    """Berhenti dengan pesan jelas bila masukan tidak layak. Mengembalikan kode EPSG data."""
    if not arcpy.Exists(jalur):
        raise ValueError("data tidak ada atau tidak bisa dibuka")
    nama_field = [f.name for f in arcpy.ListFields(jalur)]
    kurang = [f for f in FIELD_WAJIB if f not in nama_field]
    if kurang:
        raise ValueError("field hilang: " + ", ".join(kurang))
    return arcpy.Describe(jalur).spatialReference.factoryCode

for jalur in sorted(glob.glob(DATA + r"\Kiriman_Lapangan\shp\*.shp")):
    nama = os.path.basename(jalur)
    try:
        epsg = periksa(jalur)
        if epsg != 32749:
            arcpy.Project_management(jalur, "in_memory\\" + nama[:-4], arcpy.SpatialReference(32749))
            log.append((nama, "DITERIMA", "CRS %s diubah ke 32749" % epsg))
        else:
            log.append((nama, "DITERIMA", "sesuai"))
    except ValueError as e:
        # galat milik skrip kita
        log.append((nama, "DITOLAK", str(e)))
    except arcpy.ExecuteError:
        # galat dari alat geoprocessing
        log.append((nama, "GAGAL ALAT", arcpy.GetMessages(2)))
    except Exception:
        # galat Python lain
        log.append((nama, "TAK TERDUGA", str(sys.exc_info()[1])))
for baris in log:
    print("%-14s %-12s %s" % baris)
```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. [CEK]

Cek paham

1. Apa beda blok `else` dan `finally` pada `try`?
2. Mengapa `except:` kosong sebaiknya dihindari?
3. Mengapa fungsi `jalankan_aman` mengembalikan pasangan (hasil, pesan)?

Jawaban:

1. `else` hanya jalan bila tidak ada galat. `finally` selalu jalan.
2. Karena menelan semua galat, termasuk kesalahan Anda sendiri, sehingga masalah tersembunyi.
3. Agar pemanggil bisa memeriksa hasil dan membaca sebab kegagalan tanpa menulis `try` berulang kali.

Kesalahan umum

- **Menangkap galat lalu diam.** Selalu catat sebabnya ke layar atau berkas log.
- **Memperbaiki data tanpa mencatat.** Tulis di log bahwa CRS diubah atau titik digeser.
- **Memeriksa terlalu sedikit.** Berkas yang bisa dibuka belum tentu field-nya lengkap, dan koordinatnya belum tentu di area yang benar.
- **Percaya penuh pada angka "berhasil".** Jumlah fitur gabungan harus cocok dengan jumlah kiriman yang diterima.

Ringkasan dan latihan

Ringkasan: periksa masukan sebelum dipakai, tangkap galat dengan `try-except` pada jenis yang diperkirakan, dan catat hasil tiap berkas. Batch yang baik tidak berhenti karena satu berkas, dan menutup pekerjaan dengan laporan diterima, ditolak, dan curiga.

Latihan:

1. Tambahkan satu pemeriksaan baru ke Skrip 4.2: tolak berkas yang `featureCount()` bernilai 0.
2. Ubah Skrip 3.1 (NDVI ubin) agar tahan galat. Jalankan pada folder `Ortofoto_Campur`, lalu pastikan tiga ubin baik tetap diproses dan ubin rusak dicatat.

Tabel perbandingan: penanganan galat

Hal	QGIS	ArcGIS Pro	ArcMap 10.8
Galat alat	<code>QgsProcessingException</code>	<code>arcpy.ExecuteError</code>	<code>arcpy.ExecuteError</code>
Pesan galat alat	<code>str(e)</code>	<code>arcpy.GetMessages(2)</code>	<code>arcpy.GetMessages(2)</code>
Cek data sah	<code>isValid()</code>	<code>arcpy.Exists()</code> [CEK]	<code>arcpy.Exists()</code> [CEK]
Cek CRS	<code>layer.crs().authid()</code>	<code>Describe().spatialReference.factoryCode</code> [CEK]	<code>Describe().spatialReference.factoryCode</code> [CEK]
Ubah CRS	<code>native:reprojectlayer</code>	<code>arcpy.management.Project</code> [CEK]	<code>arcpy.Project_management</code> [CEK]
Catatan ke pengguna	<code>QgsMessageLog</code>	<code>arcpy.AddError,</code> <code>arcpy.AddMessage</code>	<code>arcpy.AddError,</code> <code>arcpy.AddMessage</code>

BAB 5: Ekspresi: Bahasa Rumus yang Dipakai di Mana-Mana

Studi kasus: "Kalimat laporan yang menulis dirinya sendiri"

Kepala Seksi ingin satu kalimat ringkas untuk tiap petak, misalnya "KPH Beta, 4 ha, 5 titik panas". Setiap bulan angkanya berubah, dan Anda mengetik ulang semuanya. Bulan ini Anda mencoba cara lain: menuliskan **rumusnya** sekali. QGIS yang menghitung dan menyusun kalimatnya untuk setiap petak.

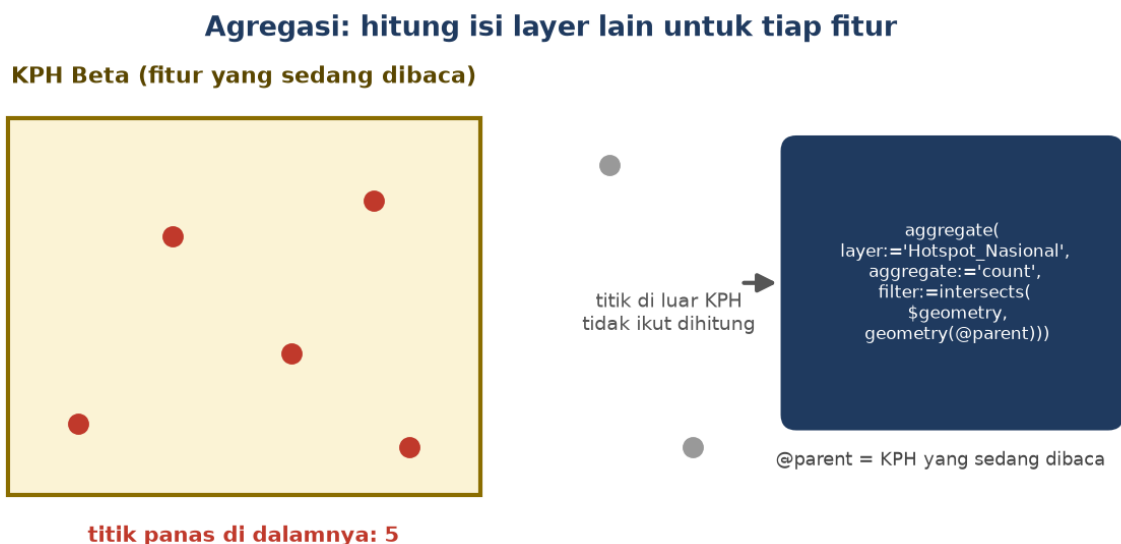
Konsep: ekspresi dalam tiga kalimat

Ekspresi adalah rumus tertulis yang dihitung QGIS untuk setiap fitur, mirip rumus di sel lembar kerja. Rumus yang sama dipakai di banyak tempat: seleksi, kalkulator field, label, atlas, dan model. **Agregasi** menambah satu kemampuan: menghitung ringkasan dari layer **lain**, misalnya jumlah titik panas yang jatuh di dalam tiap petak.

Istilah baru bab ini:

- **Ekspresi**: rumus yang dihitung per fitur.
- **Fungsi ekspresi**: perintah kecil di dalam rumus, seperti `round()` atau `upper()`.
- **Konteks**: "lingkungan" tempat rumus dihitung, yaitu fitur yang sedang dibaca dan variabel proyek.
- **Agregasi** (`aggregate`): ringkasan dari layer lain, seperti jumlah, rata-rata, atau gabungan teks.
- `@parent`: fitur yang sedang dibaca di layer luar, dipakai di dalam agregasi.

Ilustrasi 5.1: Agregasi: hitung isi layer lain untuk tiap fitur



Skema; bukan tangkapan layar

BAGIAN A | QGIS

Bagian A: Ekspresi di QGIS

A.1 Ekspresi dasar

Dua aturan tanda kutip harus dihafal. **Tanda kutip ganda** ("NAMA_KPH") menunjuk nama kolom.

Tanda kutip tunggal ('KPH Beta') menulis teks. Operator || menyambung teks. Hasil uji di QGIS

4.0.2 untuk KPH Alpha:

Ekspresi	Hasil
<code>round(\$area / 10000, 2)</code>	4
<code>upper("NAMA_KPH")</code>	KPH ALPHA
<code>replace("NAMA_KPH", 'KPH ', '')</code>	Alpha
<code>if(\$area > 50000, 'besar', 'kecil')</code>	kecil
<code>CASE WHEN \$area > 50000 THEN 'besar' ELSE 'kecil' END</code>	kecil

Gabungan teks ditulis dengan ||. Contoh yang menghasilkan KPH Alpha (4 ha):

EKSPRESI QGIS

```
"NAMA_KPH" || ' (' || round($area / 10000, 1) || ' ha)'
```

Dua jenis luas: \$area mengikuti pengaturan **ellipsoid** proyek, sedangkan area(\$geometry) selalu memakai bidang datar peta. Pada uji, proyek tanpa ellipsoid memberi 40.000 m² untuk keduanya. Setelah proyek diberi ellipsoid WGS 84, \$area menjadi 39.997,03 m² dan area(\$geometry) tetap 40.000. Untuk satuan hektare dengan dua desimal, hasilnya sama (4,00 ha). Pada pekerjaan yang butuh ketelitian, pilih satu cara dan catat.

A.2 Menghitung ekspresi dari Python

Skrip 5.1 membangun "lingkungan" (konteks) lalu menghitung ekspresi untuk tiap petak. Skrip ini juga menunjukkan dua jenis galat ekspresi: **galat tulis** (rumus tidak lengkap) dan **galat hitung** (rumus benar tetapi nilainya tidak bisa dihitung).

PYTHON (PYQGIS) 5.1

```
# [SKRIP 5.1: Mengevaluasi ekspresi QGIS dari Python, termasuk agregasi]
# Penulis: Badar Mubarak Yogaswara
from qgis.core import (QgsProject, QgsVectorLayer, QgsExpression, QgsExpressionContext,
                       QgsExpressionContextUtils, QgsCoordinateReferenceSystem)

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

proyek = QgsProject.instance()
batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
hotspot = QgsVectorLayer(DATA + "/Hotspot_Nasional.gpkg", "Hotspot_Nasional", "ogr")
```

```

survei = QgsVectorLayer(DATA + "/Survei.gpkg", "Titik_Survei", "ogr")
for lyr in (batas, hotspot, survei):
    proyek.addMapLayer(lyr)          # fungsi aggregate() mencari layer lain berdasarkan NAMA di proyek

def konteks(layer, fitur):
    """Membuat 'lingkungan' tempat ekspresi dihitung: variabel proyek, layer, dan fitur saat ini."""
    c = QgsExpressionContext()
    c.appendScopes(QgsExpressionContextUtils.globalProjectLayerScopes(layer))
    c.setFeature(fitur)
    return c

def hitung(teks, layer, fitur):
    e = QgsExpression(teks)
    if e.hasParserError():
        # salah tulis ekspresi
        return "GALAT TULIS: " + e.parserErrorString()
    nilai = e.evaluate(konteks(layer, fitur))
    if e.hasEvalError():
        # salah saat dihitung
        return "GALAT HITUNG: " + e.evalErrorString()
    return nilai

# 1. Ekspresi dasar: rumus, teks, dan pilihan (CASE)
dasar = {
    "luas (ha)": "round($area / 10000, 2)",
    "nama huruf besar": 'upper("NAMA_KPH")',
    "ukuran": "CASE WHEN $area > 50000 THEN 'besar' ELSE 'kecil' END",
}
for fitur in batas.getFeatures():
    print(fitur["NAMA_KPH"], {k: hitung(v, batas, fitur) for k, v in dasar.items()})

# 2. Agregasi: menghitung dari LAYER LAIN. @parent = fitur KPH yang sedang dibaca.
titik_panas = ("aggregate(layer='Hotspot_Nasional', aggregate='count', expression=\"ID_Titik\", \"
    \"filter:=intersects($geometry, geometry(@parent))\"")
tinggi_rata = ("round(aggregate(layer='Titik_Survei', aggregate='mean', expression=\"Tinggi_Phn\", \"
    \"filter:=intersects($geometry, geometry(@parent))\", 1)")
print("--- agregasi per KPH ---")
for fitur in batas.getFeatures():
    print("%-10s titik panas = %s | tinggi pohon rata-rata = %s" % (
        fitur["NAMA_KPH"], hitung(titik_panas, batas, fitur), hitung(tinggi_rata, batas, fitur)))

# 3. Agregasi tanpa filter: satu angka untuk seluruh layer
fitur = next(batas.getFeatures())
total = hitung("aggregate('Hotspot_Nasional', 'count', \"ID_Titik\")", batas, fitur)
print("Total titik panas (semua KPH):", total)

# 4. Ekspresi yang salah dikenali dengan aman
print(hitung("round($area / 10000, ", batas, fitur))
print(hitung("to_int('abc') + 1", batas, fitur))

# 5. Dua cara lain menghitung titik: daftar dan gabungan teks
lain = {
    "jumlah lewat overlay": "array_length(overlay_intersects('Hotspot_Nasional', \"ID_Titik\"))",
    "daftar ID": ("aggregate('Hotspot_Nasional', 'concatenate', \"ID_Titik\", \"
        \"filter:=intersects($geometry, geometry(@parent))\", concatenator=', ')",
}
for fitur in batas.getFeatures():
    print(fitur["NAMA_KPH"], {k: hitung(v, batas, fitur) for k, v in lain.items()})

# 6. Awas: $area mengikuti pengaturan ellipsoid proyek, area($geometry) memakai bidang datar peta
fitur = next(batas.getFeatures())
print("Tanpa ellipsoid :", hitung("$area", batas, fitur), hitung("area($geometry)", batas, fitur))
proyek.setCrs(QgsCoordinateReferenceSystem("EPSG:32749"))
proyek.setEllipsoid("EPSG:7030")          # WGS 84, pengaturan umum pada proyek baru
print("Dengan ellipsoid :", round(hitung("$area", batas, fitur), 2), hitung("area($geometry)", batas, fitur))

```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 5.1

```
KPH Alpha {'luas (ha)': 4.0, 'nama huruf besar': 'KPH ALPHA', 'ukuran': 'kecil'}
KPH Beta {'luas (ha)': 4.0, 'nama huruf besar': 'KPH BETA', 'ukuran': 'kecil'}
KPH Gamma {'luas (ha)': 8.0, 'nama huruf besar': 'KPH GAMMA', 'ukuran': 'besar'}
--- agregasi per KPH ---
KPH Alpha titik panas = 2 | tinggi pohon rata-rata = 11.3
KPH Beta titik panas = 5 | tinggi pohon rata-rata = 9.6
KPH Gamma titik panas = 5 | tinggi pohon rata-rata = 8.9
Total titik panas (semua KPH): 12
GALAT TULIS:
Incomplete expression. You might not have finished the full expression.
GALAT HITUNG: Cannot convert 'abc' to int
KPH Alpha {'jumlah lewat overlay': 2, 'daftar ID': 'HS-008, HS-012'}
KPH Beta {'jumlah lewat overlay': 5, 'daftar ID': 'HS-004, HS-005, HS-006, HS-009, HS-011'}
KPH Gamma {'jumlah lewat overlay': 5, 'daftar ID': 'HS-001, HS-002, HS-003, HS-007, HS-010'}
Tanpa ellipsoid : 40000.0 40000.0
Dengan ellipsoid : 39997.03 40000.0
```

Bedah kode: `QgsExpression("...")` membuat rumus dari teks. `evaluate(konteks)` menghitungnya untuk satu fitur. `hasParserError()` memeriksa salah tulis. `hasEvalError()` memeriksa salah hitung. Periksa keduanya sebelum memakai hasil. Agregasi menulis `aggregate(layer:=..., aggregate:=..., expression:=..., filter:=...)`. Tanda `:=` memberi nama pada argumen. Layer yang dirujuk **harus ada di proyek** dengan nama persis sama. Pada uji, layer yang tidak ada menghasilkan NULL dengan pesan `Cannot find layer with name or ID`.

A.3 Agregasi: menghitung dari layer lain

Perhatikan satu petak, misalnya KPH Beta. Rumusnya berbunyi: "dari semua titik panas, hitung hanya yang **bersinggungan dengan petak ini**". Bagian "petak ini" ditulis `geometry(@parent)`. Bagian "bersinggungan" ditulis `intersects($geometry, ...)`, di mana `$geometry` adalah titik panas yang sedang diperiksa.

Hasil uji: KPH Alpha 2 titik, KPH Beta 5 titik, dan KPH Gamma 5 titik. Totalnya 12 titik, sama dengan jumlah titik di layer. Dua cara lain yang memberi hasil sama:

`array_length(overlay_intersects('Hotspot_Nasional', "ID_Titik"))` untuk jumlah, dan agregasi `'concatenate'` untuk daftar ID titik.

Titik di garis batas: `intersects` menghitung titik yang tepat di garis batas sebagai milik **kedua** petak tetangga. Bila itu mengganggu, pakai `within` agar titik hanya dihitung bila benar-benar berada di dalam poligon.

A.4 Ekspresi di dalam alat Processing

Ekspresi tidak hanya untuk label. Skrip 5.2 memakainya di **Field calculator** (menambah kolom luas dan jumlah titik panas) dan di **Extract by expression** (memilih petak dengan titik panas minimal 5). Terakhir, skrip menyusun teks laporan untuk tiap petak.

PYTHON (PYQGIS) 5.2

```
# [SKRIP 5.2: Ekspresi di dalam alat Processing: Field calculator, Extract by expression, label]
# Penulis: Badar Mubarak Yogaswara
import processing
from qgis.core import (QgsProject, QgsVectorLayer, QgsExpression, QgsExpressionContext,
                      QgsExpressionContextUtils)

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

proyek = QgsProject.instance()
batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
hotspot = QgsVectorLayer(DATA + "/Hotspot_Nasional.gpkg", "Hotspot_Nasional", "ogr")
proyek.addMapLayer(batas)
proyek.addMapLayer(hotspot)          # wajib ada di proyek agar aggregate() menemukannya

# 1. Menambah dua kolom baru dengan ekspresi
tahap1 = processing.run("native:fieldcalculator", {
    "INPUT": batas, "FIELD_NAME": "LUAS_HA", "FIELD_TYPE": 0,          # 0 = desimal
    "FIELD_LENGTH": 10, "FIELD_PRECISION": 2,
    "FORMULA": "round($area / 10000, 2)", "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
tahap2 = processing.run("native:fieldcalculator", {
    "INPUT": tahap1, "FIELD_NAME": "JML_HOTSPOT", "FIELD_TYPE": 1,    # 1 = bilangan bulat
    "FIELD_LENGTH": 5, "FIELD_PRECISION": 0,
    "FORMULA": "aggregate('Hotspot_Nasional', 'count', \"ID_Titik\", \"
        intersects($geometry, geometry(@parent))\"",
    "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
tahap2.setName("Batas_KPH_Ringkas")
proyek.addMapLayer(tahap2)
print("Kolom:", [f.name() for f in tahap2.fields()])
for f in tahap2.getFeatures():
    print("  ", f["NAMA_KPH"], f["LUAS_HA"], "ha;", f["JML_HOTSPOT"], "titik panas")

# 2. Memilih fitur dengan ekspresi lewat alat
rawan = processing.run("native:extractbyexpression", {
    "INPUT": tahap2, "EXPRESSION": '"JML_HOTSPOT" >= 5', "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
print("KPH dengan titik panas >= 5:", sorted(f["NAMA_KPH"] for f in rawan.getFeatures()))

# 3. Teks laporan satu KPH, dibentuk dengan ekspresi (cara yang sama dipakai label di layout)
teks = ("LAPORAN ' || upper(\"NAMA_KPH\") || '\n' || 'Luas: ' || \"LUAS_HA\" || ' ha\n' || \"
    'Titik panas: ' || \"JML_HOTSPOT\" || ' titik'")
ekspresi = QgsExpression(teks)
for f in tahap2.getFeatures():
    c = QgsExpressionContext()
    c.appendScopes(QgsExpressionContextUtils.globalProjectLayerScopes(tahap2))
    c.setFeature(f)
    print(ekspresi.evaluate(c))
    print("---")
```

Hasil uji di QGIS 4.0.2:**HASIL UJI SKRIP 5.2**

```
Kolom: ['fid', 'NAMA_KPH', 'LUAS_HA', 'JML_HOTSPOT']
KPH Alpha 4.0 ha; 2 titik panas
KPH Beta 4.0 ha; 5 titik panas
KPH Gamma 8.0 ha; 5 titik panas
KPH dengan titik panas >= 5: ['KPH Beta', 'KPH Gamma']
LAPORAN KPH ALPHA
Luas: 4 ha
Titik panas: 2 titik
---
LAPORAN KPH BETA
Luas: 4 ha
Titik panas: 5 titik
```

```

---
LAPORAN KPH GAMMA
Luas: 8 ha
Titik panas: 5 titik
---

```

Hasilnya: kolom `LUAS_HA` dan `JML_HOTSPOT` terbentuk, KPH Beta dan KPH Gamma terpilih, dan teks laporan tersusun sendiri untuk tiap petak.

BAGIAN B | ArcGIS Pro

Bagian B: Ekspresi di ArcGIS Pro

ArcGIS Pro memakai **Arcade** untuk ekspresi bergaya QGIS. Alat **Calculate Field** menerima ekspresi Arcade, dan lewat ArcPy jenis ekspresinya ditulis "ARCADE". [CEK] Untuk menghitung titik dalam poligon, Arcade memakai `FeatureSetByName`, `Intersects`, dan `Count`. Pada Calculate Field, pencarian layer lain dilaporkan memakai `$datastore`, bukan `$map`. Bentuk persisnya perlu diuji pada geodatabase Anda. [CEK]

PYTHON (ARCPY, PYTHON 3) [CEK]

```

import arcpy
arcpy.management.CalculateField("Batas_KPH", "LUAS_HA", "Round(Area($feature, 'hectares'), 2)", "ARCADE")
arcpy.management.CalculateField("Batas_KPH", "HOTSPOT",
    "Count(Intersects($feature, FeatureSetByName($datastore, 'Hotspot_Nasional')))", "ARCADE")

```

Cara yang lebih pasti adalah **Spatial Join**, yang menambah kolom `Join_Count` (lihat Skrip 2.1-Pro). Ekspresi Python pada Calculate Field juga tersedia, misalnya `!shape.area@hectares!`. [CEK]

BAGIAN C | ArcMap 10.8

Bagian C: Ekspresi di ArcMap 10.8

ArcMap tidak punya Arcade. **Field Calculator** menerima ekspresi **VB Script** atau **Python**. Untuk luas hektare, ekspresi Python-nya `!shape.area@hectares!`. [CEK] Agregasi lintas layer **tidak ada padanan langsung** di kalkulator field. Pakai **Spatial Join** (kolom `Join_Count`) atau **Summary Statistics**, seperti pada Skrip 2.1-ArcMap.

PYTHON (ARCPY, PYTHON 2.7) [CEK]

```

import arcpy
arcpy.CalculateField_management("Batas_KPH", "LUAS_HA", "!shape.area@hectares!", "PYTHON_9.3")

```

Cek paham

1. Apa arti `@parent` di dalam aggregate?
2. Mengapa layer `Hotspot_Nasional` harus ada di proyek saat rumus agregasi dihitung?

3. Apa beda `$area` dan `area($geometry)`?

Jawaban:

1. Fitur yang sedang dibaca pada layer luar, misalnya petak KPH yang sedang dihitung.
2. Karena `aggregate` mencari layer lain berdasarkan nama di proyek. Bila tidak ketemu, hasilnya `NULL` dengan pesan galat.
3. `$area` mengikuti ellipsoid pengaturan proyek. `area($geometry)` memakai bidang datar peta.

Kesalahan umum

- **Tertukar tanda kutip.** "KPH Alpha" dibaca sebagai nama kolom dan menghasilkan galat `Field 'KPH Alpha' not found`. Teks ditulis 'KPH Alpha'.
- **Menyambung dengan nilai kosong.** `'a' || NULL` menghasilkan `NULL`, bukan `a`. Bungkus dengan `coalesce(nilai, 0)`.
- **Layer agregasi tidak ada di proyek.** Tambahkan layer ke proyek dengan nama yang sama persis.
- **Lupa memeriksa galat ekspresi.** Pakai `hasParserError()` dan `hasEvalError()` sebelum memakai hasil.

Ringkasan dan latihan

Ringkasan: ekspresi adalah rumus per fitur. Kutip ganda untuk kolom, kutip tunggal untuk teks.

`aggregate` menghitung dari layer lain dengan filter `intersects($geometry, geometry(@parent))`. Ekspresi dapat dihitung dari Python (`QgsExpression`) maupun di dalam alat (Field calculator, Extract by expression).

Latihan:

1. Tulis ekspresi yang menghasilkan "KPH Gamma: 8 ha, 5 titik panas". Hasil yang benar untuk Gamma memakai luas 8 dan jumlah titik 5.
2. Pakai `aggregate` dengan 'mean' pada `Survei.gpkg` (layer `Titik_Survei`) untuk menghitung tinggi pohon rata-rata tiap petak. Hasil pada Skrip 5.1: Alpha 11,3, Beta 9,6, dan Gamma 8,9.

Tabel perbandingan: ekspresi

Hal	QGIS	ArcGIS Pro	ArcMap 10.8
Bahasa ekspresi	Ekspresi QGIS	Arcade atau Python [CEK]	VB Script atau Python [CEK]
Kolom hitungan	Field calculator	Calculate Field [CEK]	Field Calculator [CEK]
Agregasi layer lain	<code>aggregate(...)</code>	Arcade <code>FeatureSetByName</code> [CEK] atau Spatial Join	Tidak ada padanan langsung; Spatial Join
Hitung dari Python	<code>QgsExpression</code>	<code>CalculateField</code> dengan "ARCADE" [CEK]	<code>CalculateField_management</code> [CEK]
Dipakai di label dan atlas	Ya	Label Arcade [CEK]	Tidak seperti QGIS [CEK]

BAB 6: Modeler Lanjut: Mesin dengan Tombol Pengaturan

Studi kasus: "Sempadan 20, 30, atau 50 meter?"

Kepala Seksi belum memutuskan lebar sempadan sungai. "Coba 20 meter, 30 meter, lalu 50 meter, dan tunjukkan luasnya." Anda ingin satu mesin yang menerima lebar sebagai isian. Rekan Anda juga harus bisa memakainya tanpa menyentuh kode, dan mesin itu harus menolak isian yang tidak masuk akal, misalnya lebar nol.

Konsep: model lanjut dalam tiga kalimat

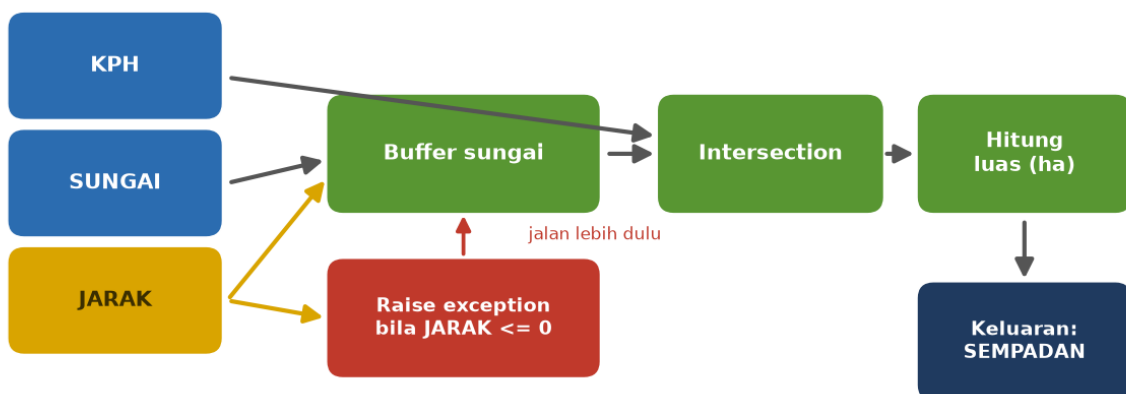
Model adalah alur kerja bergambar: kotak masukan, kotak alat, dan panah di antaranya. Model **lanjut** punya isian (parameter) yang bisa diubah, ekspresi di dalamnya, dan penjaga yang menghentikan model bila masukan tidak layak. Bayangkan mesin cuci dengan tombol pengaturan: mesinnya satu, tetapi hasilnya berbeda menurut pengaturan.

Istilah baru bab ini:

- **Masukan model:** isian yang diisi pengguna saat model dijalankan.
- **Langkah:** satu alat di dalam model.
- **Penjaga:** langkah yang memeriksa syarat dan menghentikan model bila syarat gagal.
- **Berkas model** (`.model3`): berkas tempat model disimpan.
- **Ekspor skrip:** mengubah model menjadi skrip Python.

Ilustrasi 6.1: Model: masukan, penjaga, langkah, keluaran

Model: masukan, penjaga, langkah, keluaran



Skema; bukan tangkapan layar

BAGIAN A | QGIS

Bagian A: Model Designer di QGIS

A.1 Menyusun model dengan klik

Langkah-langkah ini mengikuti dokumentasi QGIS. Letak tombol di versi Anda bisa sedikit berbeda, dan tampilan visualnya perlu Anda cocokkan. [CEK]

1. Buka **Processing ► Model Designer**. Pada versi lama, menu ini bernama Graphical Modeler. [CEK]
2. Beri nama model, misalnya "Sempadan Sungai per KPH".
3. Di panel **Inputs**, tambahkan tiga masukan: Vector Layer berjenis poligon (beri nama KPH), Vector Layer berjenis garis (SUNGAI), dan Number (JARAK) dengan nilai bawaan 30.
4. Di panel **Algorithms**, tambahkan **Buffer**. Untuk parameter Input, pilih masukan SUNGAI. Untuk Distance, pilih masukan JARAK.
5. Tambahkan **Intersection**. Masukannya KPH dan hasil Buffer.
6. Tambahkan **Field calculator** untuk kolom `LUAS_HA` dengan rumus `round($area / 10000, 2)`.
7. Beri nama keluaran akhir, lalu simpan model sebagai berkas `.model3`.
8. Jalankan dengan **Model ► Run Model...** (F5).

Hasilnya: sebuah model yang bisa dijalankan berulang dengan lebar berbeda hanya dengan mengubah satu isian.

A.2 Isian model dan ekspresi

Nilai sebuah parameter langkah dapat berasal dari empat sumber. Nama di tabel adalah nama yang dipakai oleh Python API QGIS. Nama itu sudah dipakai dan terbukti jalan pada Skrip 6.1.

Sumber nilai	Padanan di Python	Contoh
Nilai tetap	<code>fromStaticValue(30)</code>	Parameter DISSOLVE selalu benar
Masukan model	<code>fromModelParameter("JARAK")</code>	Jarak buffer diisi pengguna
Keluaran langkah lain	<code>fromChildOutput(id, "OUTPUT")</code>	Hasil Buffer menjadi masukan Intersection
Nilai hasil ekspresi	<code>fromExpression("@JARAK * 2")</code>	Satu angka dihitung dari isian model

Menurut dokumentasi QGIS, masukan model tersedia sebagai **variabel** di dalam ekspresi (misalnya `@JARAK`), dan opsi **Pre-calculated Value** membuka Expression Builder. [CEK: letak opsi di antarmuka]

Rumus per fitur bukan nilai pra-hitung: Nilai hasil ekspresi dihitung **sekali** untuk seluruh model, tanpa fitur. Rumus kolom seperti `round($area / 10000, 2)` harus berupa **nilai tetap** pada parameter `FORMULA`, karena QGIS menghitungnya sendiri untuk tiap fitur. Pada uji, rumus ini yang diberikan sebagai nilai hasil ekspresi membuat langkah gagal dengan pesan `Error encountered while running Hitung luas (ha)`.

A.3 Penjaga: menghentikan model dengan pesan jelas

Alat **Raise exception** menghentikan model dengan pesan buatan Anda. Alat ini punya dua parameter: pesan, dan syarat berupa ekspresi. Menurut keterangan alatnya, galat hanya dikeluarkan bila syarat bernilai benar. Pada model Anda, syaratnya `@JARAK <= 0`.

QGIS juga menyediakan **Conditional branch** (cabang bersyarat), yang membuat bagian model berjalan hanya bila ekspresinya benar. Kedua alat ini tidak tampil di Processing Toolbox, tetapi terdaftar di QGIS 4.0.2 dan dipakai di Model Designer. [CEK: letaknya di panel Algorithms] Cabang bersyarat diatur di antarmuka, dan perlu Anda coba sendiri.

Mengulang di QGIS: Daftar alat QGIS 4.0.2 tidak memuat alat perulangan. Untuk mengulang model dengan banyak nilai, pakai perulangan Python (Scrip 6.2) atau **Execute as batch process** (Bab 3). Bila Model Designer Anda punya fitur pengulangan, fitur itu tidak dibahas di sini. [CEK]

A.4 Membangun model lewat Python

Model hanyalah berkas yang bisa dibuat siapa saja, termasuk skrip. Skrip 6.1 membangun model yang sama dengan langkah A.1, lengkap dengan penjaga, lalu menyimpannya sebagai `.model3`. Salinan hasilnya tersedia di folder `skrip/model`.

PYTHON (PYQGIS) 6.1

```
# [SKRIP 6.1: Membangun model "Sempadan Sungai per KPH" lewat Python, simpan sebagai .model3]
# Penulis: Badar Mubarak Yogaswara
# Model ini sama dengan yang bisa Anda gambar di Graphical Modeler; ini hanya cara lain membuatnya.
import os

from qgis.core import (QgsProcessing, QgsProcessingModelAlgorithm, QgsProcessingModelChildAlgorithm,
                      QgsProcessingModelChildParameterSource as Sumber, QgsProcessingModelParameter,
                      QgsProcessingModelOutput, QgsProcessingModelChildDependency,
                      QgsProcessingParameterVectorLayer, QgsProcessingParameterNumber)

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
os.makedirs(HASIL, exist_ok=True)

m = QgsProcessingModelAlgorithm("Sempadan Sungai per KPH", "Kehutanan")

# --- Masukkan model (yang diisi pengguna saat model dijalankan)
m.addModelParameter(QgsProcessingParameterVectorLayer(
    "KPH", "Layer batas KPH", [QgsProcessing.TypeVectorPolygon]), QgsProcessingModelParameter("KPH"))
m.addModelParameter(QgsProcessingParameterVectorLayer(
    "SUNGAI", "Layer sungai", [QgsProcessing.TypeVectorLine]), QgsProcessingModelParameter("SUNGAI"))
m.addModelParameter(QgsProcessingParameterNumber(
    "JARAK", "Lebar sempadan (m)", QgsProcessingParameterNumber.Double, 30),
    QgsProcessingModelParameter("JARAK"))

def anak(alat, deskripsi):
    c = QgsProcessingModelChildAlgorithm(alat)
    c.generateChildId(m)
    c.setDescription(deskripsi)
    return c

def dari_input(nama):
    return [Sumber.fromModelParameter(nama)]

def nilai(v):
    return [Sumber.fromStaticValue(v)]
```

```

def dari_anak(c, keluaran="OUTPUT"):
    return [Sumber.fromChildOutput(c.childId(), keluaran)]

# --- Penjaga: bila JARAK tidak lebih dari 0, model berhenti dengan pesan jelas
henti = anak("native:raiseexception", "Hentikan bila jarak tidak sah")
henti.addParameterSources("MESSAGE", nilai("Lebar sempadan harus lebih dari 0 meter."))
henti.addParameterSources("CONDITION", [Sumber.fromExpression("@JARAK <= 0")])
m.addChildAlgorithm(henti)

# --- Langkah utama: baru berjalan sesudah penjaga lolos
buf = anak("native:buffer", "Penyangga sungai")
buf.addParameterSources("INPUT", dari_input("SUNGAI"))
buf.addParameterSources("DISTANCE", dari_input("JARAK"))
buf.addParameterSources("DISSOLVE", nilai(True))
buf.setDependencies([QgsProcessingModelChildDependency(henti.childId())])
m.addChildAlgorithm(buf)

iris = anak("native:intersection", "Irisan dengan KPH")
iris.addParameterSources("INPUT", dari_input("KPH"))
iris.addParameterSources("OVERLAY", dari_anak(buf))
m.addChildAlgorithm(iris)

luas = anak("native:fieldcalculator", "Hitung luas (ha)")
luas.addParameterSources("INPUT", dari_anak(iris))
luas.addParameterSources("FIELD_NAME", nilai("LUAS_HA"))
luas.addParameterSources("FIELD_TYPE", nilai(0))
luas.addParameterSources("FIELD_LENGTH", nilai(10))
luas.addParameterSources("FIELD_PRECISION", nilai(2))
# rumus dihitung per fitur, jadi diberikan sebagai nilai tetap (teks)
luas.addParameterSources("FORMULA", nilai("round($area / 10000, 2)"))
keluaran = QgsProcessingModelOutput("SEMPADAN", "Sempadan per KPH")
keluaran.setChildOutputName("OUTPUT")
luas.setModelOutputs({"SEMPADAN": keluaran})
m.addChildAlgorithm(luas)

berkas = HASIL + "/Sempadan_Sungai_per_KPH.model3"
print("Model tersimpan?", m.toFile(berkas))
print("Langkah:", [c.description() for c in m.childAlgorithms().values()])
print("Masukan:", [p.name() for p in m.parameterDefinitions()])

```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 6.1

Model tersimpan? True

Langkah: ['Penyangga sungai', 'Hitung luas (ha)', 'Irisan dengan KPH', 'Hentikan bila jarak tidak sah']

Masukan: ['KPH', 'SUNGAI', 'JARAK', 'sempadan_per_kph']

Bedah kode: `QgsProcessingModelChildAlgorithm("native:buffer")` membuat satu langkah. `addParameterSources` mengisi parameternya dengan salah satu dari empat sumber di tabel A.2. `setDependencies` mengatur urutan: langkah Penyangga baru berjalan setelah Penjaga lolos. Nama kunci keluaran model dibentuk dari nama keluaran. Lihat daftarnya dengan `model.parameterDefinitions()` (pada uji: `sempadan_per_kph`).

A.5 Menjalankan model dari Python

Skrip 6.2 memuat berkas model, lalu menjalankannya tiga kali dengan lebar 20, 30, dan 50 meter. Terakhir, skrip mencoba lebar 0 untuk menguji penjaga.

PYTHON (PYQGIS) 6.2

```
# [SKRIP 6.2: Menjalankan model .model3 dari Python, berulang dengan jarak berbeda]
# Penulis: Badar Mubarak Yogaswara
# Prasyarat: jalankan SKRIP 6.1 dulu (atau simpan model dari Graphical Modeler dengan nama sama).
import processing
from qgis.core import QgsVectorLayer, QgsProcessingModelAlgorithm, QgsProcessingException

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"

batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
sungai = QgsVectorLayer(DATA + "/Sungai.gpkg", "Sungai", "ogr")

# 1. Muat model dari berkas
model = QgsProcessingModelAlgorithm()
print("Model terbaca?", model.fromFile(HASIL + "/Sempadan_Sungai_per_KPH.model3"))
model.initAlgorithm({})
print("Nama model:", model.displayName())
print("Parameter:", [p.name() for p in model.parameterDefinitions()])

# 2. Jalankan berulang dengan tiga lebar sempadan. Model yang sama, angka berbeda.
for jarak in (20, 30, 50):
    hasil = processing.run(model, {
        "KPH": batas, "SUNGAI": sungai, "JARAK": jarak,
        "sempadan_per_kph": "TEMPORARY_OUTPUT"})
    lyr = hasil["sempadan_per_kph"]
    total = sum(f["LUAS_HA"] for f in lyr.getFeatures())
    print("Lebar %2d m -> %d potongan, total %.2f ha" % (jarak, lyr.featureCount(), total))

# 3. Jarak yang tidak sah: model berhenti dan memberi pesan
try:
    processing.run(model, {"KPH": batas, "SUNGAI": sungai, "JARAK": 0,
        "sempadan_per_kph": "TEMPORARY_OUTPUT"})
    print("Jarak 0: model berjalan (tidak seharusnya)")
except QgsProcessingException as e:
    print("Jarak 0: model berhenti ->", str(e).replace("\n", " | "))
```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 6.2

```
Model terbaca? True
Nama model: Sempadan Sungai per KPH
Parameter: ['JARAK', 'KPH', 'SUNGAI', 'sempadan_per_kph']
Lebar 20 m -> 3 potongan, total 2.75 ha
Lebar 30 m -> 3 potongan, total 4.07 ha
Lebar 50 m -> 3 potongan, total 6.55 ha
Jarak 0: model berhenti -> Lebar sempadan harus lebih dari 0 meter.
```

Angka dapat dicek silang dengan Bab 2: pada lebar 30 m, jumlah luas tiga petak adalah $1,18 + 1,39 + 1,50 = 4,07$ ha, sama dengan hasil Skrip 2.2. Dengan lebar 0, penjaga menghentikan model dan pesannya terbaca jelas.

A.6 Dari model ke skrip

Menu **Model ► Export as Script Algorithm...** mengubah model menjadi skrip algoritma Python yang bisa diedit. Bab ini tidak membahasnya lebih jauh. Skrip algoritma menjadi bahan seri M4.

BAGIAN B | ArcGIS Pro

Bagian B: ModelBuilder di ArcGIS Pro

ModelBuilder di Pro punya **iterator**, yaitu elemen perulangan. Daftarnya antara lain Iterate Feature Selection, Iterate Rasters, Iterate Field Values, Iterate Files, Iterate Datasets, Iterate Feature Classes, Iterate Fields, dan For. Menurut dokumentasi Esri, hanya satu iterator yang dapat dipakai per model. Hasil iterator dipakai sebagai **variabel sisipan** (misalnya %Name%) pada nama keluaran. [CEK]

Untuk isian, klik kanan variabel lalu jadikan **Parameter**. Untuk syarat, ModelBuilder punya **Preconditions** dan alat **If Value Is**. Model bisa diekspor menjadi skrip Python lewat menu **Export**. [CEK]

Padanan model Bab 6: Iterate Feature Selection mengulang per petak, sedangkan Buffer dan Intersect sama dengan alat QGIS. Skrip padanannya adalah Skrip 2.1-Pro dan 3.1-Pro. Alat **Raise exception** tidak ada padanan langsung; penjaga dibuat dengan alat **If Value Is** atau pemeriksaan di skrip. [CEK]

BAGIAN C | ArcMap 10.8

Bagian C: ModelBuilder di ArcMap 10.8

ModelBuilder di ArcMap juga punya iterator, misalnya **Iterate Feature Selection** dan **Iterate Rasters**. Hasilnya dipakai sebagai variabel sisipan seperti %Name% pada nama keluaran. Parameter model dibuat dengan klik kanan variabel. Ekspor ke skrip Python ada di menu **Model**. [CEK]

Skrip padanan model adalah Skrip 2.1-ArcMap dan 3.1-ArcMap. Penjaga dibuat lewat pemeriksaan di skrip (Skrip 4.1-ArcMap), karena padanan **Raise exception** tidak ditemukan di dokumentasi yang dibaca. [CEK]

Cek paham

1. Apa guna alat **Raise exception** di dalam model?
2. Bagaimana nilai masukan model dipakai di dalam ekspresi?
3. Mengapa menjalankan model dari Python berguna?

Jawaban:

1. Menghentikan model dengan pesan jelas bila syarat (ekspresi) bernilai benar, misalnya jarak tidak sah.
2. Sebagai variabel dengan awalan @, misalnya @JARAK.
3. Agar model bisa diulang dengan banyak nilai dan disambung dengan skrip lain, seperti proyek terpadu di Bab 7.

Kesalahan umum

- **Rumus kolom dijadikan nilai pra-hitung.** Berikan rumus per fitur sebagai nilai tetap, bukan hasil ekspresi.
- **Salah nama kunci hasil.** Nama keluaran model dibentuk dari nama keluaran, bukan selalu OUTPUT. Periksa dengan `parameterDefinitions()`.
- **Mencari Raise exception di Processing Toolbox.** Alat ini tersembunyi dari toolbox. Cari di Model Designer. [CEK]
- **Mengisi jarak pada layer berderajat.** Satuan jarak mengikuti CRS layer, jadi pastikan metrik.

Ringkasan dan latihan

Ringkasan: model lanjut punya masukan, ekspresi, dan penjaga. Model disimpan sebagai `.model3`, dibangun lewat klik atau Python, dan dijalankan dari Python dengan `processing.run(model, parameter)`. Rumus per fitur diberikan sebagai nilai tetap.

Latihan:

1. Ubah Skrip 6.1 agar model menolak JARAK di atas 200 (syarat `@JARAK <= 0 OR @JARAK > 200`). Jalankan dengan jarak 250 dan baca pesannya.
2. Pada Skrip 6.2, tambahkan jarak 100 m. Perhatikan apakah luas sempadan tiap petak tumbuh sebanding dengan lebarnya.

Tabel perbandingan: model alur kerja

Hal	QGIS	ArcGIS Pro	ArcMap 10.8
Nama alat	Model Designer (dulu Graphical Modeler) [CEK]	ModelBuilder	ModelBuilder
Isian model	Masukan model	Parameter model [CEK]	Parameter model [CEK]
Perulangan dalam model	Tidak ada alat perulangan di daftar QGIS 4.0.2	Iterator (satu per model) [CEK]	Iterator [CEK]
Penjaga	Raise exception, Conditional branch	Preconditions, If Value Is [CEK]	Preconditions [CEK]
Ekspor ke skrip	Export as Script Algorithm	Export ke Python [CEK]	Export ke Python [CEK]
Jalankan dari Python	<code>processing.run(model, {...})</code> (diuji)	Alat model dipanggil seperti alat biasa [CEK]	Alat model dipanggil seperti alat biasa [CEK]

BAB 7: Bab Jembatan ke Master: Proyek Mini Terpadu dan Ceklis Masuk M1

Studi kasus: "Laporan awal bulan, satu skrip"

Awal bulan tiba. Kepala Seksi minta laporan lengkap: kiriman lapangan yang sudah digabung, luas sempadan sungai dan titik panas per petak, serta kondisi vegetasi dari citra dan drone. Tahun lalu pekerjaan ini memakan dua hari. Hari ini Anda menjalankannya dalam beberapa detik. Sambil menunggu hasil, Anda membaca daftar isi **Masterclass GIS Modern (M1)**: citra cloud, survei lapangan, drone, LiDAR, klasifikasi objek, dan atlas. Anda ingin tahu: apakah bekal Anda cukup?

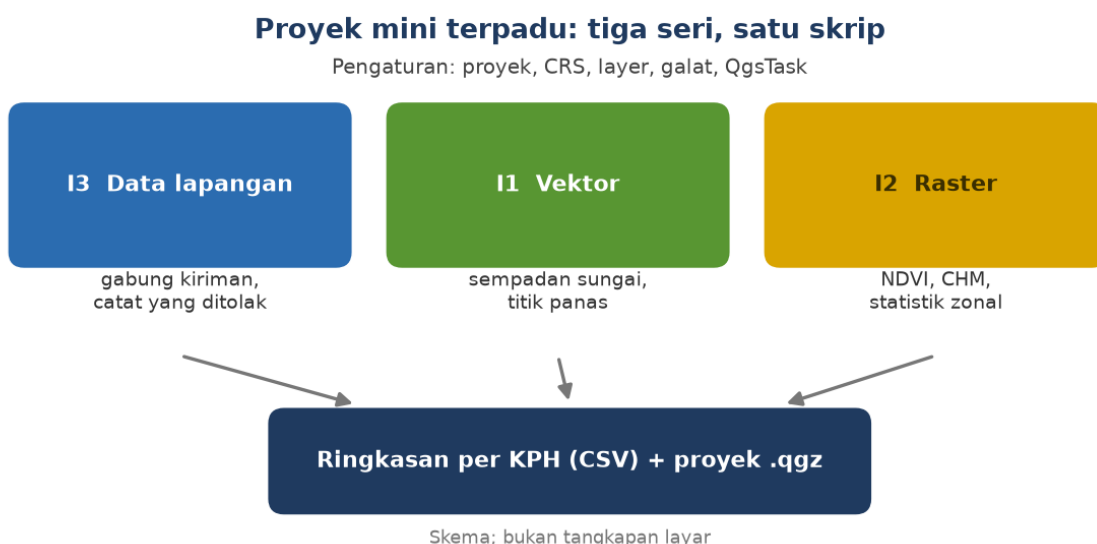
Konsep: jembatan dalam tiga kalimat

Bab Jembatan adalah proyek kecil yang memadukan semua yang Anda pelajari di kelas Intermediate, ditambah pemeriksaan bekal sebelum masuk kelas Master. Seperti ujian praktik sebelum turun ke jalan raya, proyek ini menguji tangan Anda, bukan hanya ingatan Anda. Setelah lulus ceklisnya, Anda siap menyeberang ke M1.

Istilah baru bab ini:

- **Proyek terpadu**: satu skrip yang memadukan beberapa seri.
- **Keluaran permanen**: hasil yang disimpan di GeoPackage atau berkas, bukan hanya di memori.
- **Ceklis masuk**: daftar kemampuan yang M1 anggap sudah Anda kuasai.

Ilustrasi 7.1: Proyek mini terpadu



BAGIAN A | QGIS

Bagian A: Jembatan di QGIS

A.1 Rancangan proyek

Proyek ini memakai data paket `paket-i4`. Tiap bagian memakai kemampuan dari satu seri dan satu bab di buku ini.

Seri	Pekerjaan	Alat dan kemampuan	Bab di buku ini
I3	Gabung enam kiriman lapangan, tolak yang rusak, ubah CRS, tandai titik di luar area	<code>QgsVectorLayer</code> , <code>isValid</code> , <code>try-except</code> , <code>reprojectlayer</code> , <code>mergevectorlayers</code>	Bab 1 dan 4
I1	Sempadan sungai 30 m per KPH, titik panas, titik survei per KPH	<code>buffer</code> , <code>intersection</code> , <code>fieldcalculator</code> , <code>aggregate</code>	Bab 2 dan 5
I2	NDVI, tinggi tajuk (CHM), rata-rata per KPH	<code>rastercalc</code> , <code>zonalstatisticsfb</code>	Bab 2
Penutup	Ringkasan CSV, GeoPackage hasil, dan proyek <code>.qgz</code>	<code>native:package</code> , <code>QgsProject.write</code>	Bab 1

A.2 Skrip terpadu

Skrip 7.1 berjalan dari atas ke bawah. Bagian 0 menyiapkan pembantu: fungsi memuat layer yang menolak masukan rusak, dan fungsi `jalankan` yang menyingkat pemanggilan alat. Setelah itu tiap bagian seri mengerjakan tugasnya.

PYTHON (PYQGIS) 7.1

```
# [SKRIP 7.1: Proyek mini terpadu I1-I3: dari kiriman lapangan sampai ringkasan per KPH]
# Penulis: Badar Mubarak Yogaswara
# Keluaran di folder HASIL: Proyek_Terpada.qgz, Hasil_Terpada.gpkg, ringkasan_kph.csv, log_kiriman.csv.
import csv
import glob
import os
import processing
from qgis.core import QgsProject, QgsVectorLayer, QgsRasterLayer, QgsCoordinateReferenceSystem

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
os.makedirs(HASIL, exist_ok=True)

CRS = "EPSG:32749"
SEMENTARA = "TEMPORARY_OUTPUT"
proyek = QgsProject.instance()
proyek.removeAllMapLayers()
proyek.setCrs(QgsCoordinateReferenceSystem(CRS))
proyek.setTitle("KPH Contoh - proyek terpadu")

# ----- bagian 0: pembantu
def muat_vektor(jalur, nama):
    lyr = QgsVectorLayer(jalur, nama, "ogr")
    if not lyr.isValid():
```

```

        raise ValueError("vektor tidak bisa dibuka: " + jalur)
    return lyr

def muat_raster(jalur, nama):
    lyr = QgsRasterLayer(jalur, nama, "gdal")
    if not lyr.isValid():
        raise ValueError("raster tidak bisa dibuka: " + jalur)
    return lyr

def jalankan(alat, **parameter):
    """Menjalankan alat Processing lalu mengembalikan isi kunci OUTPUT."""
    return processing.run(alat, parameter)["OUTPUT"]

def hitung_kolom(layer, nama, tipe, rumus):
    """Menambah satu kolom dengan ekspresi. tipe: 0 = desimal, 1 = bilangan bulat."""
    return jalankan("native:fieldcalculator", INPUT=layer, FIELD_NAME=nama, FIELD_TYPE=tipe,
                    FIELD_LENGTH=10, FIELD_PRECISION=2 if tipe == 0 else 0, FORMULA=rumus,
                    OUTPUT=SEMENTARA)

batas = muat_vektor(DATA + "/Batas_KPH.gpkg", "Batas_KPH")
sungai = muat_vektor(DATA + "/Sungai.gpkg", "Sungai")
hotspot = muat_vektor(DATA + "/Hotspot_Nasional.gpkg", "Hotspot_Nasional")
citra = muat_raster(DATA + "/Citra_KPH.tif", "Citra_KPH")
dsm = muat_raster(DATA + "/DSM_Drone.tif", "DSM")
dtm = muat_raster(DATA + "/DTM_Drone.tif", "DTM")
for lyr in (batas, sungai, hotspot, citra):
    proyek.addMapLayer(lyr)

# ----- bagian I3: data lapangan
tujuan_crs = QgsCoordinateReferenceSystem(CRS)
diterima, log = [], []
for jalur in sorted(glob.glob(DATA + "/Kiriman_Lapangan/*.gpkg")):
    nama = os.path.basename(jalur)
    try:
        lyr = muat_vektor(jalur, nama)
        kurang = [f for f in ("Kondisi", "Tinggi_Phn", "Foto") if lyr.fields().indexOf(f) < 0]
        if kurang:
            raise ValueError("field hilang: " + ", ".join(kurang))
        if lyr.crs().authid() != CRS:
            lyr = jalankan("native:reprojectlayer", INPUT=lyr, TARGET_CRS=tujuan_crs, OUTPUT=SEMENTARA)
        diterima.append(lyr)
        log.append((nama, "DITERIMA"))
    except Exception as e:
        log.append((nama, "DITOLAK: %s" % e))

gabung = jalankan("native:mergevectorlayers", LAYERS=diterima, CRS=tujuan_crs, OUTPUT=SEMENTARA)
# Tiap berkas membawa nomor fid sendiri, jadi setelah digabung ada fid kembar. Buang kolom itu;
# GeoPackage akan memberi nomor baru yang unik saat disimpan.
survei = jalankan("native:deletecolumn", INPUT=gabung, COLUMN=["fid"], OUTPUT=SEMENTARA)
survei.setName("Titik_Survei_Gabungan")
proyek.addMapLayer(survei)
with open(HASIL + "/log_kiriman.csv", "w", newline="", encoding="utf-8") as f:
    csv.writer(f).writerows([("berkas", "status")] + log)
ditolak = sum(1 for _, s in log if s != "DITERIMA")
print("Kiriman:", len(log) - ditolak, "diterima,", ditolak, "ditolak;",
      "titik gabungan =", survei.featureCount())

# ----- bagian I1: analisis vektor
sempadan = jalankan("native:buffer", INPUT=sungai, DISTANCE=30, DISSOLVE=True, OUTPUT=SEMENTARA)
sempadan.setName("Sempadan_30m")
proyek.addMapLayer(sempadan)
irisasi = jalankan("native:intersection", INPUT=batas, OVERLAY=sempadan, INPUT_FIELDS=["NAMA_KPH"],
                  OVERLAY_FIELDS=[], OUTPUT=SEMENTARA)
sempadan_ha = {}

```

```

for f in irisan.getFeatures():
    sempadan_ha[f["NAMA_KPH"]] = sempadan_ha.get(f["NAMA_KPH"], 0) + f.geometry().area() / 10000

# Kolom baru lewat ekspresi: luas, titik panas, titik survei (agregasi dari layer lain)
cacah = "aggregate('%s', 'count', \"%s\", intersects($geometry, geometry(@parent)))"
ringkas = hitung_kolom(batas, "LUAS_HA", 0, "round($area / 10000, 2)")
ringkas = hitung_kolom(ringkas, "HOTSPOT", 1, cacah % ("Hotspot_Nasional", "ID_Titik"))
ringkas = hitung_kolom(ringkas, "TITIK_SURVEI", 1, cacah % ("Titik_Survei_Gabungan", "Foto"))
ringkas.setName("Ringkasan_KPH")
proyek.addMapLayer(ringkas)

# ----- bagian I2: analisis raster
ndvi_jalur = jalankan("native:rastercalc", LAYERS=[citra], OUTPUT=HASIL + "/NDVI_KPH.tif",
    EXPRESSION='("Citra_KPH@4" - "Citra_KPH@1") / ("Citra_KPH@4" + "Citra_KPH@1")')
chm_jalur = jalankan("native:rastercalc", LAYERS=[dsm, dtm], OUTPUT=HASIL + "/CHM.tif",
    EXPRESSION='("DSM@1" - "DTM@1")')
ndvi = muat_raster(ndvi_jalur, "NDVI_KPH")
chm = muat_raster(chm_jalur, "CHM")
proyek.addMapLayer(ndvi)
proyek.addMapLayer(chm)

def rata(raster, awalan):
    z = jalankan("native:zonalstatisticsfb", INPUT=batas, INPUT_RASTER=raster, RASTER_BAND=1,
        COLUMN_PREFIX=awalan, STATISTICS=[2], OUTPUT=SEMENTARA)
    return {f["NAMA_KPH"]: f[awalan + "mean"] for f in z.getFeatures()}

ndvi_rata, chm_rata = rata(ndvi, "n_"), rata(chm, "c_")

# ----- bagian akhir: tabel dan proyek
judul = ["NAMA_KPH", "LUAS_HA", "SEMPADAN_HA", "HOTSPOT", "TITIK_SURVEI", "NDVI_RATA", "CHM_RATA_M"]
baris = []
for f in ringkas.getFeatures():
    n = f["NAMA_KPH"]
    baris.append([n, f["LUAS_HA"], round(sempadan_ha.get(n, 0), 2), f["HOTSPOT"], f["TITIK_SURVEI"],
        round(ndvi_rata[n], 3), round(chm_rata[n], 2)])
with open(HASIL + "/ringkasan_kph.csv", "w", newline="", encoding="utf-8") as f:
    w = csv.writer(f)
    w.writerow(judul)
    w.writerows(baris)
print("%-10s %-7s %-12s %-8s %-13s %-10s %-11s" % tuple(judul))
for b in baris:
    print("%-10s %-7.2f %-12.2f %-8d %-13d %-10.3f %-11.2f" % tuple(b))

# Layer sementara hilang saat QGIS ditutup. Simpan yang penting ke satu GeoPackage,
# lalu ganti layer sementara di proyek dengan layer dari GeoPackage itu.
paket = HASIL + "/Hasil_Terpadu.gpkg"
processing.run("native:package", {"LAYERS": [survei, sempadan, ringkas], "OUTPUT": paket,
    "OVERWRITE": True})

for lyr in (survei, sempadan, ringkas):
    nama = lyr.name()
    proyek.removeMapLayer(lyr.id())
    proyek.addMapLayer(muat_vektor(paket + "|layername=" + nama, nama))
print("Proyek tersimpan?", proyek.write(HASIL + "/Proyek_Terpadu.qgz"))
print("Isi proyek:", sorted(l.name() for l in proyek.mapLayers().values()))

```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 7.1

Kiriman: 4 diterima, 2 ditolak; titik gabungan = 18

NAMA_KPH	LUAS_HA	SEMPADAN_HA	HOTSPOT	TITIK_SURVEI	NDVI_RATA	CHM_RATA_M
KPH Alpha	4.00	1.18	2	6	0.604	2.60
KPH Beta	4.00	1.39	5	4	0.431	2.69

```
KPH Gamma      8.00      1.50      5      7      0.329      2.56
Proyek tersimpan? True
Isi proyek: ['Batas_KPH', 'CHM', 'Citra_KPH', 'Hotspot_Nasional', 'NDVI_KPH', 'Ringkasan_KPH', 'Sempadan_30m',
'Sungai', 'Titik_Survei_Gabungan']
```

Seluruh angka dapat dicek silang. Luas petak 4, 4, dan 8 ha sesuai data. Sempadan 1,18, 1,39, dan 1,50 ha sama dengan Bab 2. Titik panas 2, 5, dan 5 sama dengan Bab 5. NDVI 0,604, 0,431, dan 0,329 sama dengan Bab 2. Titik survei 6, 4, dan 7 sesuai pembuat data: tim A mengirim 6 titik di KPH Alpha, tim B 4 titik di KPH Beta (satu titiknya di luar area), serta tim C dan D 4 + 3 titik di KPH Gamma. Total 17 titik di dalam area, ditambah 1 di luar, yaitu 18 titik hasil gabungan.

Bedah kode: `def jalankan(alat, **parameter)` menerima parameter bernama sebanyak apa pun dan mengumpulkannya dalam satu kamus. Karena itu `jalankan("native:buffer", INPUT=sungai, DISTANCE=30)` sama dengan `processing.run("native:buffer", {...})` ["OUTPUT"], tetapi lebih pendek. `native:deletecolumn` membuang kolom `fid`. Tiap kiriman membawa nomor `fid` sendiri. Setelah digabung, nomor itu kembar, dan `GeoPackage` menolaknya saat menyimpan dengan galat `UNIQUE constraint failed`. Buang kolomnya dan biarkan `GeoPackage` memberi nomor baru. `native:package` menyimpan beberapa layer sekaligus ke satu `GeoPackage`. Lalu layer sementara di proyek diganti dengan layer dari `GeoPackage` itu, supaya proyek `.qgz` tetap utuh saat dibuka lagi.

Sebagai bukti, proyek `Proyek_Terpadu.qgz` yang tersimpan dibuka ulang, dan kesembilan layer di dalamnya terbaca sah. Hasil lengkanya dicatat di berkas catatan seri.

A.3 Versi latar belakang

Pada data besar, pekerjaan ini bisa berjalan lama. Skrip 7.2 memindahkan bagian beratnya ke `QgsTask`. Bagian yang menyentuh proyek, yaitu menambah layer ke peta dan menulis laporan, dikerjakan di fungsi selesai.

PYTHON (PYQGIS) 7.2

```
# [SKRIP 7.2: Ringkasan per KPH sebagai tugas latar belakang (QgsTask)]
# Penulis: Badar Mubarak Yogaswara
# Pola: pekerjaan berat di fungsi tugas (latar belakang); proyek dan tabel di fungsi selesai (utama).
import csv
import os
import processing
from qgis.core import (QgsApplication, QgsTask, QgsMessageLog, Qgis, QgsProject,
                      QgsVectorLayer, QgsRasterLayer)

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
os.makedirs(HASIL, exist_ok=True)
SEMENTARA = "TEMPORARY_OUTPUT"

def jalankan(alat, **parameter):
    return processing.run(alat, parameter)["OUTPUT"]

def hitung_ringkasan(task, data, hasil):
    """Latar belakang. Hanya membaca berkas dan menjalankan alat; tidak menyentuh QgsProject."""
    batas = QgsVectorLayer(data + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
    sungai = QgsVectorLayer(data + "/Sungai.gpkg", "Sungai", "ogr")
```

```

hotspot = QgsVectorLayer(data + "/Hotspot_Nasional.gpkg", "Hotspot_Nasional", "ogr")
citra = QgsRasterLayer(data + "/Citra_KPH.tif", "Citra_KPH", "gdal")
dsm = QgsRasterLayer(data + "/DSM_Drone.tif", "DSM", "gdal")
dtm = QgsRasterLayer(data + "/DTM_Drone.tif", "DTM", "gdal")

# tahap 1 (0-25%): sempadan sungai
sem = jalankan("native:buffer", INPUT=sungai, DISTANCE=30, DISSOLVE=True, OUTPUT=SEMENTARA)
iris = jalankan("native:intersection", INPUT=batas, OVERLAY=sem, INPUT_FIELDS=["NAMA_KPH"],
                OVERLAY_FIELDS=[], OUTPUT=SEMENTARA)

sem_ha = {}
for f in iris.getFeatures():
    sem_ha[f["NAMA_KPH"]] = sem_ha.get(f["NAMA_KPH"], 0) + f.geometry().area() / 10000
task.setProgress(25)
if task.isCanceled():
    return None

# tahap 2 (25-50%): titik panas
hit = jalankan("native:countpointsinpolygon", POLYGONS=batas, POINTS=hotspot, FIELD="HOTSPOT",
                OUTPUT=SEMENTARA)
titik = {f["NAMA_KPH"]: f["HOTSPOT"] for f in hit.getFeatures()}
task.setProgress(50)
if task.isCanceled():
    return None

# tahap 3 (50-100%): raster
ndvi = jalankan("native:rastercalc", LAYERS=[citra], OUTPUT=hasil + "/NDVI_KPH.tif",
                EXPRESSION='("Citra_KPH@4" - "Citra_KPH@1") / ("Citra_KPH@4" + "Citra_KPH@1")')
chm = jalankan("native:rastercalc", LAYERS=[dsm, dtm], OUTPUT=hasil + "/CHM.tif",
                EXPRESSION='"DSM@1" - "DTM@1"')
task.setProgress(75)

def rata(jalur, awalan):
    raster = QgsRasterLayer(jalur, "r", "gdal")
    z = jalankan("native:zonalstatisticsfb", INPUT=batas, INPUT_RASTER=raster, RASTER_BAND=1,
                COLUMN_PREFIX=awalan, STATISTICS=[2], OUTPUT=SEMENTARA)
    return {f["NAMA_KPH"]: f[awalan + "mean"] for f in z.getFeatures()}

n, c = rata(ndvi, "n_"), rata(chm, "c_")
task.setProgress(100)
hasil_baris = []
for f in batas.getFeatures():
    k = f["NAMA_KPH"]
    hasil_baris.append([k, round(f.geometry().area() / 10000, 2), round(sem_ha.get(k, 0), 2),
                        titik[k], round(n[k], 3), round(c[k], 2)])

return hasil_baris

def selesai(exception, baris=None):
    """Program utama. Aman menyentuh proyek, tabel, dan tampilan."""
    if exception is not None:
        QgsMessageLog.logMessage("Ringkasan gagal: %s" % exception, "Ringkasan KPH", Qgs.Critical)
        return
    if baris is None:
        QgsMessageLog.logMessage("Ringkasan dibatalkan", "Ringkasan KPH", Qgs.Warning)
        return
    with open(HASIL + "/ringkasan_latar.csv", "w", newline="", encoding="utf-8") as f:
        w = csv.writer(f)
        w.writerow(["NAMA_KPH", "LUAS_HA", "SEMPADAN_HA", "HOTSPOT", "NDVI_RATA", "CHM_RATA_M"])
        w.writerows(baris)
    for nama in ("NDVI_KPH", "CHM"):
        QgsProject.instance().addMapLayer(QgsRasterLayer(HASIL + "/" + nama + ".tif", nama, "gdal"))
    for b in baris:
        print("%-10s luas %5.2f ha | sempadan %4.2f ha | titik panas %d | NDVI %3f | CHM %2f m"
              % tuple(b))
    QgsMessageLog.logMessage("Ringkasan selesai: %d KPH" % len(baris), "Ringkasan KPH", Qgs.Success)

tugas = QgsTask.fromFunction("Ringkasan per KPH", hitung_ringkasan, on_finished=selesai,

```

```

data=DATA, hasil=HASIL)
QgsApplication.taskManager().addTask(tugas)
print("Tugas berjalan di latar belakang; QGIS tetap bisa dipakai.")

```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 7.2

Tugas berjalan di latar belakang; QGIS tetap bisa dipakai.

KPH Alpha	luas	4.00 ha		sempadan	1.18 ha		titik panas	2		NDVI	0.604		CHM	2.60 m
KPH Beta	luas	4.00 ha		sempadan	1.39 ha		titik panas	5		NDVI	0.431		CHM	2.69 m
KPH Gamma	luas	8.00 ha		sempadan	1.50 ha		titik panas	5		NDVI	0.329		CHM	2.56 m

Angkanya sama dengan Skrip 7.1. Itu bukti bahwa memindahkan pekerjaan ke latar belakang tidak mengubah hasil.

A.4 Peta celah: apa yang M1 pakai, dan di mana Anda belajar

Ilustrasi 7.2: Menutup celah sebelum masuk Masterclass

Menutup celah sebelum masuk Masterclass



Skema; bukan tangkapan layar

Tabel berikut memetakan kebutuhan skrip dan ekspresi di M1 ke bab buku ini. Isi M1 tidak berubah. Bila pembahasan sebuah topik memang ada di M1, tabel menyebutnya.

Bab M1	Yang dipakai skripnya	Dipelajari di
Bab 1	<code>QgsProject.instance()</code> , <code>QgsRasterLayer(url, nama, "gdal")</code> , <code>mapLayersByName(...)[0]</code>	Bab 1 buku ini
Bab 1	<code>processing.run(...)</code> dengan <code>TEMPORARY_OUTPUT</code> , <code>gdal:clipprasterbymasklayer</code> , <code>native:rastercalc</code>	Bab 2 dan Skrip 3.2
Bab 1	Pengenalan STAC, COG, dan awalan <code>/vsicurl/</code>	M1 Bab 1 sendiri
Bab 2	<code>native:joinattributesbylocation</code> , perulangan fitur, hitung persen cocok	Bab 4 (Skrip 4.2)
Bab 3	CHM dari DSM dikurangi DTM, <code>QgsTask.fromFunction</code> , batch folder	Bab 2 dan 3
Bab 4	Point cloud dan alat PDAL	M1 Bab 4 sendiri
Bab 5	Statistik zonal	Bab 2 (A.4); klasifikasi OBIA dan Random Forest di M1 sendiri
Bab 6	Model dengan Conditional branch, ekspresi aggregate dengan <code>@parent</code> di label atlas	Bab 5 dan 6; layout dan atlas dari seri B4 dan M1

Satu catatan praktis. Skrip contoh di internet atau buku lain kadang memakai ID alat atau bentuk keluaran lama. Bila sebuah skrip gagal, periksa dua hal lebih dulu: nama ID alat (Skrip 2.1) dan jenis keluaran sementara (tabel di Bab 2, A.2). Kedua hal itu penyebab gagal yang paling sering ditemukan pada uji.

A.5 Ceklis masuk M1

Centang dalam hati, atau di kertas. Tiap butir berarti: Anda bisa melakukannya **tanpa melihat buku**, minimal pada data latihan.

Dari seri sebelumnya (konsep):

- Saya bisa menentukan zona UTM dan mengganti CRS proyek. (Seri B1)
- Saya paham band dan rumus NDVI. (Seri B3 dan I2)
- Saya bisa membuat GeoPackage, domain nilai, dan formulir survei. (Seri B1 dan I3)
- Saya bisa menjalankan spatial join dan statistik zonal dengan klik. (Seri I1 dan I2)
- Saya bisa membuat layout peta sederhana. (Seri B4)

Dari buku ini (praktik, tiap butir ada skripnya):

- Saya bisa membuka Python Console dan menjalankan skrip dari editor. (Bab 1)
- Saya bisa memuat layer vektor dan raster lewat skrip, lalu mengambilnya dengan nama. (Bab 1)
- Saya bisa menjalankan alat Processing dengan `processing.run` dan memakai `TEMPORARY_OUTPUT`. (Bab 2)
- Saya tahu bahwa keluaran raster sementara berupa teks jalur, dan bagaimana membungkusnya menjadi layer. (Bab 2)

- Saya bisa menulis perulangan untuk banyak berkas atau fitur. (Bab 3)
- Saya bisa memindahkan pekerjaan berat ke `QgsTask`. (Bab 3)
- Saya bisa menangkap galat dan membuat laporan diterima atau ditolak. (Bab 4)
- Saya bisa menulis ekspresi dasar, termasuk `aggregate` dengan `@parent`. (Bab 5)
- Saya bisa menyusun model dengan isian dan penjaga, lalu menjalankannya. (Bab 6)

Uji otomatis. Skrip 7.3 menguji sepuluh kemampuan di atas sekaligus. Tiap pemeriksaan mencetak LULUS atau GAGAL beserta alasannya. Skor akhir muncul setelah tugas latar belakang terakhir selesai.

PYTHON (PYQGIS) 7.3

```
# [SKRIP 7.3: Uji kesiapan otomatis: sepuluh keterampilan yang dipakai Masterclass Seri 1 (M1)]
# Penulis: Badar Mubarak Yogaswara
# Tiap pemeriksaan mencetak LULUS atau GAGAL beserta alasannya.
# Skor akhir muncul setelah tugas latar belakang (pemeriksaan 10) selesai.
import os
import re
import time
import processing
from qgis.core import (QgsApplication, QgsTask, QgsProject, QgsVectorLayer, QgsRasterLayer,
                       QgsExpression, QgsExpressionContext, QgsExpressionContextUtils,
                       QgsProcessingException)

DATA = "D:/Latihan/paket-i4"
HASIL = "D:/Latihan/hasil_i4"
os.makedirs(HASIL, exist_ok=True)

proyek = QgsProject.instance()
proyek.removeAllMapLayers()
catatan = [] # daftar (nama, lulus?, alasan)
NDVI = '("Citra_KPH@4" - "Citra_KPH@1") / ("Citra_KPH@4" + "Citra_KPH@1")'
CACAH = ("aggregate('Hotspot_Nasional', 'count', \"ID_Titik\", \"
    \"intersects($geometry, geometry(@parent))\")")

def periksa(nama, fungsi):
    try:
        alasan = fungsi()
        catatan.append((nama, True, alasan or ""))
    except Exception as e: # pemeriksaan yang gagal dicatat; skrip tidak berhenti
        catatan.append((nama, False, "%s: %s" % (type(e).__name__, e)))
    n, ok, alasan = catatan[-1]
    print("%-46s %s %s" % (n, "LULUS" if ok else "GAGAL", alasan))

def tegaskan(syarat, pesan):
    if not syarat:
        raise AssertionError(pesan)

batas = QgsVectorLayer(DATA + "/Batas_KPH.gpkg", "Batas_KPH", "ogr")
hotspot = QgsVectorLayer(DATA + "/Hotspot_Nasional.gpkg", "Hotspot_Nasional", "ogr")
citra = QgsRasterLayer(DATA + "/Citra_KPH.tif", "Citra_KPH", "gdal")

def konteks(fitur):
    c = QgsExpressionContext()
    c.appendScopes(QgsExpressionContextUtils.globalProjectLayerScopes(batas))
    c.setFeature(fitur)
    return c
```

```

def k1():
    for lyr in (batas, hotspot, citra):
        tegaskan(lyr.isValid(), lyr.name() + " tidak sah")
        proyek.addMapLayer(lyr)
    tegaskan(len(proyek.mapLayers()) == 3, "jumlah layer tidak 3")

def k2():
    tegaskan(proyek.mapLayersByName("Batas_KPH")[0].featureCount() == 3, "Batas_KPH bukan 3 fitur")
    r = proyek.mapLayersByName("Citra_KPH")[0]
    tegaskan(r.bandCount() == 4 and r.rasterUnitsPerPixelX() == 2.0, "raster tidak sesuai")
    tegaskan(proyek.mapLayersByName("Tidak_Ada") == [], "nama tak ada harus memberi daftar kosong")
    return "band=%d, piksel=%.0f m" % (r.bandCount(), r.rasterUnitsPerPixelX())

def k3():
    v = processing.run("native:buffer", {"INPUT": batas, "DISTANCE": 10,
                                         "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
    tegaskan(isinstance(v, QgsVectorLayer) and v.featureCount() == 3, "bukan layer vektor")
    r = processing.run("native:rastercalc", {"EXPRESSION": '"Citra_KPH@4" - "Citra_KPH@1"',
                                             "LAYERS": [citra], "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
    tegaskan(isinstance(r, str) and os.path.exists(r), "keluaran raster bukan jalur berkas")
    return "vektor -> layer, raster -> jalur teks"

def k4():
    jalur = processing.run("native:rastercalc", {"EXPRESSION": NDVI, "LAYERS": [citra],
                                                "OUTPUT": "TEMPORARY_OUTPUT"})["OUTPUT"]
    st = processing.run("native:rasterlayerstatistics",
                       {"INPUT": QgsRasterLayer(jalur, "ndvi", "gdal"), "BAND": 1})
    tegaskan(-1 <= st["MIN"] <= st["MAX"] <= 1, "NDVI di luar -1..1")
    return "NDVI %.2f sampai %.2f" % (st["MIN"], st["MAX"])

def k5():
    e = QgsExpression(CACAH)
    hasil = {f["NAMA_KPH"]: e.evaluate(konteks(f)) for f in batas.getFeatures()}
    benar = {f["NAMA_KPH"]: sum(1 for h in hotspot.getFeatures()
                               if h.geometry().intersects(f.geometry())) for f in batas.getFeatures()}
    tegaskan(len(hasil) == 3, "agregasi tidak menghasilkan 3 nilai: %s" % hasil)
    tegaskan(hasil == benar, "agregasi %s beda dari hitungan manual %s" % (hasil, benar))
    return str(hasil)

def k6():
    e = QgsExpression("'Titik api: ' || " + CACAH + " || ' titik'")
    beta = [x for x in batas.getFeatures() if x["NAMA_KPH"] == "KPH Beta"][0]
    teks = e.evaluate(konteks(beta))
    tegaskan(not e.hasEvalError() and teks == "Titik api: 5 titik", "teks label salah: %s" % teks)
    return teks

def k7():
    try:
        processing.run("native:buffer", {"INPUT": DATA + "/tidak_ada.gpkg", "DISTANCE": 5,
                                         "OUTPUT": "TEMPORARY_OUTPUT"})
    except QgsProcessingException:
        return "galat alat tertangkap"
    raise AssertionError("seharusnya gagal")

def k8():
    lyr = QgsVectorLayer(DATA + "/Kiriman_Lapangan/tim_e.gpkg", "rusak", "ogr")
    tegaskan(not lyr.isValid(), "berkas rusak dianggap sah")
    return "isValid() = False untuk berkas rusak"

def k9():

```

```

berkas = HASIL + "/uji_kesiapan.qgz"
tegaskan(proyek.write(berkas), "gagal menulis proyek")
proyek.clear()
tegaskan(len(proyek.mapLayers()) == 0, "proyek belum kosong")
tegaskan(proyek.read(berkas) and len(proyek.mapLayers()) == 3, "proyek tidak terbaca utuh")
return "tuliskan, kosongkan, baca ulang: 3 layer"

daftar = [("1. Muat vektor dan raster [Bab 1]", k1), ("2. Ambil layer dengan nama [Bab 1]", k2),
          ("3. processing.run sementara [Bab 2]", k3), ("4. Raster calculator NDVI [Bab 2]", k4),
          ("5. Agregasi dan @parent [Bab 5]", k5), ("6. Teks label dari ekspresi [Bab 5]", k6),
          ("7. Menangkap galat alat [Bab 4]", k7), ("8. Mendeteksi berkas rusak [Bab 4]", k8),
          ("9. Simpan dan buka proyek [Bab 1]", k9)]
for nama, fungsi in daftar:
    periksa(nama, fungsi)

# 10. Tugas latar belakang. Hasilnya tiba belakangan, jadi skor akhir dicetak di selesai().
def kerja(task):
    total = 0
    for i in range(1, 6):
        if task.isCanceled():
            return None
        time.sleep(0.05)
        total += i
        task.setProgress(i * 20)
    return total

def selesai(exception, nilai=None):
    ok = exception is None and nilai == 15
    catatan.append(("10. QgsTask sederhana [Bab 3]", ok, "hasil tugas = %s" % nilai))
    print("%-46s %s hasil tugas = %s" % (catatan[-1][0], "LULUS" if ok else "GAGAL", nilai))
    lulus = sum(1 for _, o, _ in catatan if o)
    gagal = sorted({re.search(r"\[(Bab \d)\]", n).group(1) for n, o, _ in catatan if not o})
    akhir = "Siap masuk M1." if not gagal else "Ulangi: " + ", ".join(gagal) + "."
    print("SKOR: %d dari %d. %s" % (lulus, len(catatan), akhir))

tugas_uji = QgsTask.fromFunction("Uji tugas latar belakang", kerja, on_finished=selesai)
QgsApplication.taskManager().addTask(tugas_uji)

```

Hasil uji di QGIS 4.0.2:

HASIL UJI SKRIP 7.3

1. Muat vektor dan raster [Bab 1]	LULUS
2. Ambil layer dengan nama [Bab 1]	LULUS band=4, piksel=2 m
3. processing.run sementara [Bab 2]	LULUS vektor -> layer, raster -> jalur teks
4. Raster calculator NDVI [Bab 2]	LULUS NDVI -0.02 sampai 0.84
5. Agregasi dan @parent [Bab 5]	LULUS {'KPH Alpha': 2, 'KPH Beta': 5, 'KPH Gamma': 5}
6. Teks label dari ekspresi [Bab 5]	LULUS Titik api: 5 titik
7. Menangkap galat alat [Bab 4]	LULUS galat alat tertangkap
8. Mendeteksi berkas rusak [Bab 4]	LULUS isValid() = False untuk berkas rusak
9. Simpan dan buka proyek [Bab 1]	LULUS tuliskan, kosongkan, baca ulang: 3 layer
10. QgsTask sederhana [Bab 3]	LULUS hasil tugas = 15

SKOR: 10 dari 10. Siap masuk M1.

Bila ada pemeriksaan yang GAGAL, skrip menyebut bab yang perlu diulang. Pada uji, nama satu berkas data sengaja dibuat salah, dan skor turun menjadi 4 dari 10 dengan pesan Ulangi: Bab 1, Bab 2, Bab 5. Skor 10 dari 10 berarti bekal teknis Anda cukup. Skor itu tidak menggantikan latihan nyata di data Anda sendiri.

A.6 Yang akan Anda pelajari di M1

Beberapa hal memang baru di M1, dan Anda tidak perlu menguasainya sekarang: membaca citra dari cloud (STAC dan COG), plugin STAC API Browser, peta dasar XYZ Tiles, sinkronisasi lapangan lewat QField atau Mergin Maps, pencarian puncak pohon, point cloud, Orfeo Toolbox untuk klasifikasi objek, dan atlas laporan. Kemampuan di buku ini menjadi fondasi untuk semuanya.

BAGIAN B | ArcGIS Pro

Bagian B: Jembatan di ArcGIS Pro

Proyek terpadu versi ArcPy memakai tiga alat Bab 2 (Buffer, Intersect, Spatial Join), dua perhitungan raster (NDVI dan CHM), dan Zonal Statistics as Table. Penjaganya adalah blok `try-except-finally` dari Bab 4. Skrip menulis ringkasan CSV.

PYTHON (ARCPY, PYTHON 3)

```
# [SKRIP 7.1-Pro: Proyek mini terpadu dengan ArcPy: ringkasan per KPH]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi Anda. Butuh Spatial Analyst. Data Shapefile dipakai agar sejajar dengan Bagian
# QGIS.
import csv
import arcpy
from arcpy.sa import Raster, Float, ZonalStatisticsAsTable

DATA = r"D:\Latihan\paket-i4"
HASIL = r"D:\Latihan\hasil_i4"
arcpy.env.workspace = DATA + r"\shp"
arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")

try:
    # I1: sempadan 30 m dan titik panas
    arcpy.analysis.Buffer("Sungai.shp", r"memory\sem", "30 Meters", dissolve_option="ALL")
    arcpy.analysis.Intersect(["Batas_KPH.shp", r"memory\sem"], r"memory\iris")
    arcpy.analysis.SpatialJoin("Batas_KPH.shp", "Hotspot_Nasional.shp", r"memory\ringkas",
                                "JOIN_ONE_TO_ONE", "KEEP_ALL", match_option="INTERSECT")

    # I2: NDVI dan CHM, lalu rata-rata per KPH
    nir = Float(Raster(DATA + r"\Citra_KPH.tif\Band_4"))
    merah = Float(Raster(DATA + r"\Citra_KPH.tif\Band_1"))
    ndvi = (nir - merah) / (nir + merah)
    chm = Raster(DATA + r"\DSM_Drone.tif") - Raster(DATA + r"\DTM_Drone.tif")
    ZonalStatisticsAsTable("Batas_KPH.shp", "NAMA_KPH", ndvi, r"memory\z_ndvi", "DATA", "MEAN")
    ZonalStatisticsAsTable("Batas_KPH.shp", "NAMA_KPH", chm, r"memory\z_chm", "DATA", "MEAN")

    def tabel(jalur, kolom):
        with arcpy.da.SearchCursor(jalur, ["NAMA_KPH", kolom]) as k:
            return {n: v for n, v in k}

    sempadan = {}
    with arcpy.da.SearchCursor(r"memory\iris", ["NAMA_KPH", "SHAPE@AREA"]) as k:
        for n, luas in k:
            sempadan[n] = sempadan.get(n, 0) + luas / 10000
    n_rata, c_rata = tabel(r"memory\z_ndvi", "MEAN"), tabel(r"memory\z_chm", "MEAN")
    with open(HASIL + r"\ringkasan_kph_pro.csv", "w", newline="") as f:
        w = csv.writer(f)
        w.writerow(["NAMA_KPH", "LUAS_HA", "SEMPADAN_HA", "HOTSPOT", "NDVI_RATA", "CHM_RATA_M"])
        with arcpy.da.SearchCursor(r"memory\ringkas", ["NAMA_KPH", "SHAPE@AREA", "Join_Count"]) as k:
            for n, luas, jml in k:
                w.writerow([n, round(luas / 10000, 2), round(sempadan.get(n, 0), 2), jml,
                            round(n_rata[n], 3), round(c_rata[n], 2)])
    print("Ringkasan tersimpan.")
```

```
except arcpy.ExecuteError:
    print("Alat gagal:", arcpy.GetMessages(2))
finally:
    arcpy.CheckInExtension("Spatial")
```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. Penggabungan kiriman lapangan (Bagian I3) tidak diulang di sini, karena sudah ada di Skrip 4.1-Pro. Ketersediaan `Join_Count` dan pembacaan `Band_n` perlu diuji pada Pro Anda. [CEK]

Bagian B di M1 juga memakai ArcPy pada beberapa bab. Bekal dari buku ini cukup untuk membacanya: `cursor`, `alat geoprocessing`, `ExecuteError`, dan `ruang kerja memory`.

BAGIAN C | ArcMap 10.8

Bagian C: Jembatan di ArcMap 10.8

Versi ArcMap memakai nama alat lama dan Python 2.7. Bentuknya sejajar dengan Bagian B.

PYTHON (ARCPY, PYTHON 2.7)

```
# -*- coding: utf-8 -*-
# [SKRIP 7.1-ArcMap: Proyek mini terpadu dengan ArcPy: ringkasan per KPH (Python 2.7)]
# Penulis: Badar Mubarak Yogaswara
# [CEK] Cocokkan dengan versi Anda. Butuh Spatial Analyst.
import csv
import arcpy
from arcpy.sa import Raster, Float, ZonalStatisticsAsTable

DATA = r"D:\Latihan\paket-i4"
HASIL = r"D:\Latihan\hasil_i4"
arcpy.env.workspace = DATA + r"\shp"
arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")

try:
    # I1: sempadan 30 m dan titik panas
    arcpy.Buffer_analysis("Sungai.shp", r"in_memory\sem", "30 Meters", dissolve_option="ALL")
    arcpy.Intersect_analysis(["Batas_KPH.shp", r"in_memory\sem"], r"in_memory\iris")
    arcpy.SpatialJoin_analysis("Batas_KPH.shp", "Hotspot_Nasional.shp", r"in_memory\ringkas",
                               "JOIN_ONE_TO_ONE", "KEEP_ALL", match_option="INTERSECT")

    # I2: NDVI dan CHM, lalu rata-rata per KPH
    nir = Float(Raster(DATA + r"\Citra_KPH.tif\Band_4"))
    merah = Float(Raster(DATA + r"\Citra_KPH.tif\Band_1"))
    ndvi = (nir - merah) / (nir + merah)
    chm = Raster(DATA + r"\DSM_Drone.tif") - Raster(DATA + r"\DTM_Drone.tif")
    ZonalStatisticsAsTable("Batas_KPH.shp", "NAMA_KPH", ndvi, r"in_memory\z_ndvi", "DATA", "MEAN")
    ZonalStatisticsAsTable("Batas_KPH.shp", "NAMA_KPH", chm, r"in_memory\z_chm", "DATA", "MEAN")

    def tabel(jalur, kolom):
        with arcpy.da.SearchCursor(jalur, ["NAMA_KPH", kolom]) as k:
            return dict((n, v) for n, v in k)

    sempadan = {}
    with arcpy.da.SearchCursor(r"in_memory\iris", ["NAMA_KPH", "SHAPE@AREA"]) as k:
        for n, luas in k:
            sempadan[n] = sempadan.get(n, 0) + luas / 10000.0
    n_rata, c_rata = tabel(r"in_memory\z_ndvi", "MEAN"), tabel(r"in_memory\z_chm", "MEAN")
    with open(HASIL + r"\ringkasan_kph_arcmap.csv", "wb") as f:
        # Python 2: mode "wb" untuk csv
        w = csv.writer(f)
```

```

w.writerow(["NAMA_KPH", "LUAS_HA", "SEMPADAN_HA", "HOTSPOT", "NDVI_RATA", "CHM_RATA_M"])
with arcpy.da.SearchCursor(r"in_memory\ringkas", ["NAMA_KPH", "SHAPE@AREA", "Join_Count"]) as k:
    for n, luas, jml in k:
        w.writerow([n, round(luas / 10000.0, 2), round(sembadan.get(n, 0), 2), jml,
                    round(n_rata[n], 3), round(c_rata[n], 2)])
print("Ringkasan tersimpan.")
except arcpy.ExecuteError:
    print("Alat gagal: %s" % arcpy.GetMessages(2))
finally:
    arcpy.CheckInExtension("Spatial")

```

Perlu dicek: Skrip ini perlu Anda coba dulu pada data latihan. Catatan Python 2.7: berkas CSV dibuka dengan mode "wb", dan pembagian memakai 10000.0. [CEK]

ArcMap tidak punya padanan langsung `QgsTask` dan `aggregate`. Padanan yang tidak ada ditulis terus terang di tabel perbandingan, seperti di M1.

Cek paham

1. Mengapa kolom `fid` dibuang sebelum menyimpan hasil gabungan ke `GeoPackage`?
2. Mengapa layer sementara diganti dengan layer dari `GeoPackage` sebelum proyek disimpan?
3. Pada skrip `QgsTask`, bagian mana yang dikerjakan di fungsi selesai dan mengapa?

Jawaban:

1. Setiap kiriman membawa nomor `fid` sendiri sehingga setelah digabung ada nomor kembar. `GeoPackage` menolak nomor kembar saat menyimpan.
2. Layer sementara hilang saat QGIS ditutup. Layer dari `GeoPackage` tetap ada, jadi proyek bisa dibuka lagi dengan utuh.
3. Menambah layer ke proyek dan menulis laporan. Bagian ini menyentuh proyek, sedangkan latar belakang tidak boleh menyentuhnya.

Kesalahan umum

- **Proyek disimpan dengan layer sementara.** Saat dibuka lagi, layer itu kosong atau hilang. Simpan ke `GeoPackage` dulu.
- **Mengabaikan titik yang ditandai curiga.** Titik di luar area biasanya salah ketik koordinat. Periksa sebelum dianalisis.
- **Menyentuh proyek dari latar belakang.** Pindahkan ke fungsi selesai.
- **Mengira skor 10 dari 10 berarti ahli.** Skor hanya menandakan bekal dasar sudah cukup untuk melanjutkan.

Ringkasan dan latihan

Ringkasan: proyek terpadu memadukan I3 (kiriman lapangan), I1 (analisis vektor), dan I2 (analisis raster) dalam satu skrip dengan pemeriksaan, laporan, dan penyimpanan permanen. Skrip yang sama dapat berjalan di latar belakang. Ceklis masuk M1 memuat sepuluh kemampuan, dan Skrip 7.3 mengujinya otomatis.

Latihan:

1. Tambahkan ke Skrip 7.1 kolom `TEKS_LAPORAN` yang berisi kalimat seperti "KPH Beta: 4 ha, 5 titik panas, NDVI 0,43". Gunakan ekspresi dengan `||`.
2. Ganti sempadan 30 m menjadi isian di atas skrip (satu variabel `LEBAR_SEMPADAN`), lalu jalankan dengan 20 m. Total sempadan seharusnya 2,75 ha.
3. Jalankan Skrip 7.3. Rusak sengaja satu hal (misalnya ubah `DATA` ke folder yang salah) dan lihat pemeriksaan mana yang GAGAL.

Tabel perbandingan: kemampuan masuk M1

Kemampuan	QGIS	ArcGIS Pro	ArcMap 10.8
Muat dan ambil layer	<code>QgsProject, mapLayersByName</code>	<code>arcpy.mp, listLayers</code> [CEK]	<code>arcpy.mapping, ListLayers</code> [CEK]
Keluaran sementara	<code>TEMPORARY_OUTPUT</code>	<code>memory\nama</code> [CEK]	<code>in_memory\nama</code> [CEK]
Perulangan batch	<code>for + glob</code>	<code>for + ListRasters</code> [CEK]	<code>for + ListRasters</code> [CEK]
Tugas latar belakang	<code>QgsTask</code> (diuji)	Tidak ada padanan langsung [CEK]	Tidak ada padanan langsung [CEK]
Penanganan galat	<code>try-except + QgsProcessingException</code>	<code>try-except + ExecuteError</code> [CEK]	<code>try-except + ExecuteError</code> [CEK]
Ekspresi agregasi	<code>aggregate(...)</code>	Arcade atau Spatial Join [CEK]	Spatial Join [CEK]
Model alur kerja	Model Designer	ModelBuilder [CEK]	ModelBuilder [CEK]

Lanjut belajar

Seri berikutnya adalah **M1: Masterclass GIS Modern**. Pindai kode QR ini untuk membuka halaman seri tersebut dan unduhannya.



Glosarium

Glosarium ini menjelaskan istilah teknis yang dipakai di buku ini, diurutkan menurut abjad. Bila Anda menemukan istilah yang belum jelas, cari di sini, lalu buka bab yang tertulis di akhir entri.

Agregasi (aggregation): ringkasan yang dihitung dari layer lain di dalam ekspresi, misalnya jumlah titik panas yang jatuh di dalam tiap petak. *Bab 5.*

Arcade: bahasa ekspresi milik ArcGIS Pro, mirip ekspresi QGIS. Ia dipakai, misalnya, pada alat Calculate Field. *Bab 5.*

ArcPy: paket Python milik Esri untuk memberi perintah kepada ArcGIS Pro dan ArcMap lewat skrip. *Bab 1.*

Band (band): satu lapisan nilai di dalam raster. Pada citra buku ini, band 1 adalah Merah dan band 4 adalah NIR, dan dalam ekspresi band ditulis "`NamaLayer@nomorBand`". *Pendahuluan, Bab 2.*

Batch (batch): pekerjaan yang sama dijalankan berulang pada banyak masukan. Resepnya ditulis sekali, lalu perulangan menjalankannya untuk tiap berkas. *Bab 3.*

Berkas model (`.model3`): berkas tempat model QGIS disimpan. Berkas ini bisa dibuat lewat klik atau lewat Python, lalu dijalankan lagi kapan saja. *Bab 6.*

Buffer (penyangga): alat yang membuat jalur dengan jarak tertentu di sekeliling objek, misalnya jalur 30 meter di kiri-kanan sungai. *Bab 2.*

CHM (canopy height model): raster tinggi tajuk, yaitu DSM dikurangi DTM. *Bab 2.*

Conditional branch (cabang bersyarat): alat model QGIS yang membuat sebagian model berjalan hanya bila ekspresinya benar. *Bab 6.*

CRS (coordinate reference system): sistem koordinat sebuah data. Ia menentukan satuan hitungan, misalnya meter pada UTM dan derajat pada EPSG:4326. *Bab 1.*

CSV: berkas teks berisi tabel, satu baris untuk satu data. Berkas ini mudah dibuka di lembar kerja. *Bab 1.*

Daftar (list): kumpulan nilai berurutan di Python. Nomor urutnya dimulai dari 0. *Bab 1.*

DSM (digital surface model): raster tinggi permukaan, yaitu tinggi tanah beserta benda di atasnya. *Pendahuluan, Bab 2.*

DTM (digital terrain model): raster tinggi tanah saja. *Pendahuluan, Bab 2.*

Ekspor skrip (export): perintah yang mengubah model menjadi skrip Python yang bisa diedit. *Bab 6.*

Ekspresi (expression): rumus tertulis yang dihitung per fitur, mirip rumus di sel lembar kerja. Rumus yang sama dipakai untuk seleksi, kalkulator field, label, dan model. *Bab 5.*

Elipsoid (ellipsoid): bentuk bumi yang dipakai untuk menghitung ukuran. Pada pengaturan proyek, `$area` mengikutinya, sedangkan `area($geometry)` memakai bidang datar peta. *Bab 5.*

EPSG: kode angka untuk sebuah CRS. Contohnya EPSG:32749 untuk UTM zona 49S. *Pendahuluan, Bab 1.*

ExecuteError (`arcpy.ExecuteError`): galat yang dipicu ArcPy bila sebuah alat geoprocessing gagal. Pesannya dibaca dengan `arcpy.GetMessages(2)`. *Bab 4.*

Field (kolom atribut): satu kolom di tabel atribut layer vektor, misalnya `NAMA_KPH`. *Pendahuluan, Bab 1.*

Field calculator (kalkulator field): alat yang mengisi kolom dengan hasil sebuah ekspresi. *Bab 5.*

Fitur (feature): satu baris di layer vektor, yaitu satu objek peta. *Bab 1.*

Fungsi (function): resep bernama di Python yang bisa dipakai berulang kali. *Bab 1.*

Fungsi ekspresi: perintah kecil di dalam rumus, seperti `round()` atau `upper()`. *Bab 5.*

Fungsi selesai (`on_finished`): fungsi yang dipanggil di program utama setelah sebuah tugas latar belakang berakhir. Di sinilah hasil boleh ditambahkan ke proyek. *Bab 3.*

Galat (exception): sinyal dari Python bahwa ada yang gagal. Tanpa penanganan, skrip langsung berhenti. *Bab 4.*

GeoPackage: berkas data (ekstensi `.gpkg`) yang dapat memuat beberapa layer sekaligus. Hasil penting disimpan di sini agar tidak hilang. *Pendahuluan, Bab 7.*

ID alat: nama unik sebuah alat Processing, misalnya `native:buffer`. *Bab 2.*

Indentasi (indentation): baris yang menjorok ke dalam di Python. Ia menandai isi sebuah blok, dan empat spasi sudah cukup. *Bab 1.*

Intersection (irisan): alat yang memotong satu layer dengan layer lain dan hanya menyisakan bagian yang saling tumpang. *Bab 2.*

isValid: pemeriksaan pada layer yang menjawab benar atau salah. Layer yang gagal dibuka tidak menimbulkan galat, sehingga hasil ini harus diperiksa sendiri. *Bab 1.*

Iterator (iterator): elemen perulangan di ModelBuilder ArcGIS. Hanya satu iterator yang dapat dipakai per model. *Bab 6.*

Kamus (dictionary): pasangan kunci dan nilai di Python, seperti daftar isi berisi nomor halaman. *Bab 1.*

Keluaran permanen: hasil yang disimpan di GeoPackage atau berkas, bukan hanya di memori. *Bab 7.*

Keluaran sementara (`TEMPORARY_OUTPUT`): hasil yang disimpan di memori atau folder sementara, bukan di berkas pilihan Anda. Vektor kembali sebagai layer, sedangkan raster kembali sebagai teks jalur. *Bab 2.*

Konsol Python (Python Console): jendela di QGIS tempat perintah Python diketik atau dijalankan. *Bab 1.*

Konteks (context): lingkungan tempat ekspresi dihitung, yaitu fitur yang sedang dibaca dan variabel proyek. *Bab 5.*

Kursor (cursor): alat ArcPy untuk membaca baris tabel satu per satu, misalnya `arcpy.da.SearchCursor`. Ia padanan `getFeatures()` di QGIS. *Bab 1.*

Latar belakang (background): jalur kerja terpisah yang tidak membekukan layar QGIS. *Bab 3.*

Layer (layer): satu data di proyek, bisa vektor atau raster. *Bab 1.*

Log (log): catatan hasil tiap langkah yang bisa dibaca belakangan. *Bab 4.*

Masukan model (model input): isian yang diisi pengguna saat model dijalankan. *Bab 6.*

Model Designer: jendela QGIS untuk menyusun model alur kerja bergambar. Pada versi lama namanya Graphical Modeler. *Bab 6.*

ModelBuilder: jendela ArcGIS Pro dan ArcMap untuk menyusun model alur kerja bergambar. *Bab 6.*

NDVI (normalized difference vegetation index): angka kehijauan vegetasi, dihitung dengan $(\text{NIR} - \text{Merah}) \div (\text{NIR} + \text{Merah})$. *Bab 2.*

Nilai pra-hitung (pre-calculated value): nilai parameter model yang dihitung sekali dari sebuah ekspresi untuk seluruh model, tanpa fitur. Rumus per fitur tidak boleh diberikan dengan cara ini. *Bab 6.*

Parameter (parameter): isian sebuah alat, seperti masukan, jarak, dan keluaran. *Bab 2.*

@parent: fitur yang sedang dibaca di layer luar, dipakai di dalam agregasi. *Bab 5.*

Penjaga (guard): langkah di model atau skrip yang memeriksa syarat dan menghentikan pekerjaan bila syarat gagal. *Bab 6.*

Penyedia data (provider): kata yang memberi tahu QGIS cara membaca data. "ogr" untuk vektor dan "gdal" untuk raster. *Bab 1.*

Perulangan (loop, `for`): perintah yang mengulang pekerjaan untuk tiap isi daftar. *Bab 1.*

Piksel (pixel): kotak terkecil dalam raster. Tiap kotak menyimpan satu angka. *Pendahuluan, Bab 1.*

Pola berkas (glob): cara mencari banyak berkas sekaligus, misalnya *.tif. *Bab 3.*

processing.run: perintah Python yang menjalankan sebuah alat Processing. Anda memberi ID alat dan kamus parameter, lalu menerima kamus hasil. *Bab 2.*

Proyek (project): proyek QGIS yang sedang terbuka, dilihat dari Python lewat `QgsProject`. *Bab 1.*

Proyek terpadu: satu skrip yang memadukan beberapa seri. *Bab 7.*

PyQGIS: cara Python berbicara dengan QGIS. *Bab 1.*

Python: bahasa perintah tertulis yang dipakai di buku ini. *Bab 1.*

QgsProcessingException: galat khusus yang dikeluarkan alat Processing saat gagal. *Bab 4.*

QgsTask: tugas latar belakang milik QGIS. Pekerjaan berat dikirim ke sana agar QGIS tetap bisa dipakai. *Bab 3.*

Raise exception: alat model QGIS yang menghentikan model dengan pesan buatan Anda bila syarat bernilai benar. *Bab 6.*

Rantai alat: susunan alat yang keluaran satu alat menjadi masukan alat berikutnya. *Bab 2.*

Raster (raster): data berbentuk kotak-kotak, satu angka tiap kotak, seperti citra dan DSM. *Bab 1.*

Raster calculator: alat hitung pada raster. Di QGIS alat ini dipakai dengan ID `native:rastercalc`. *Bab 2.*

Reproyeksi (reproject, ubah CRS): mengubah sebuah layer ke CRS lain, misalnya dari EPSG:4326 ke EPSG:32749. *Bab 4.*

Ruang kerja memory (memory): tempat hasil antara di ArcGIS Pro yang disimpan di RAM. Ia padanan `TEMPORARY_OUTPUT`, dan di ArcMap namanya `in_memory`. *Bab 2.*

Shapefile: format data vektor yang dipakai pada salinan data untuk ArcGIS. *Pendahuluan, Bab 1.*

Skrip (script): berkas teks berisi perintah Python, biasanya berakhiran `.py`. *Bab 1.*

Spatial Analyst: ekstensi ArcGIS untuk analisis raster. Ekstensi ini harus aktif untuk alat raster pada Bagian B dan C. *Pendahuluan, Bab 2.*

Spatial join: alat yang menggabungkan atribut dua layer menurut letaknya. Di ArcGIS ia menambah kolom `Join_Count` untuk menghitung titik. *Bab 2.*

Statistik zonal (zonal statistics): ringkasan nilai raster di dalam tiap poligon. *Bab 2.*

Titik panas (hotspot): titik pada layer `Hotspot_Nasional` yang jumlahnya dihitung per petak KPH. *Pendahuluan, Bab 2.*

try-except: blok Python yang mencoba sesuatu dan menangkap galatnya. *Bab 4.*

Ubin (tile): satu lembar potongan foto udara. Satu area besar terdiri atas banyak ubin. *Pendahuluan, Bab 3.*

UTM (Universal Transverse Mercator): sistem koordinat berukuran meter. Di buku ini dipakai zona 49S. *Pendahuluan, Bab 1.*

Validasi (validation): pemeriksaan masukan sebelum dipakai. *Bab 4.*

Variabel (variable): kotak bernama yang menyimpan nilai. *Bab 1.*

Variabel sisipan (inline variable): nama dalam tanda persen, misalnya `%Name%`, yang menyisipkan hasil iterator ke nama keluaran di ModelBuilder. *Bab 6.*

Vektor (vector): data berbentuk titik, garis, atau poligon, dengan tabel atribut. *Bab 1.*

Daftar Pustaka

Daftar ini memuat sumber yang dipakai menyusun buku ini. Semua tautan dibuka dan dicek pada 7 Oktober 2026.

Dokumentasi resmi

- Esri. (t.t.). *Calculate Field (Data Management Tools)*. ArcGIS Pro documentation, Esri. <https://doc.esri.com/en/arcgis-pro/latest/tool-reference/data-management/calculate-field.html>.
- Esri. (t.t.). *Error handling with Python*. ArcGIS Pro documentation, Esri. <https://doc.esri.com/en/arcgis-pro/latest/arcpy/get-started/error-handling-with-python.html>.
- Esri. (2021). *Introduction to arcpy.mapping*. ArcMap documentation, Esri. <http://desktop.arcgis.com/en/arcmap/latest/analyze/arcpy-mapping/introduction-to-arcpy-mapping.htm>.
- Esri. (t.t.). *Iterators*. ArcGIS Pro documentation, Esri. <https://doc.esri.com/en/arcgis-pro/latest/help/analysis/geoprocessing/modelbuilder/iterators-for-looping.html>.
- Esri. (t.t.). *Spatial Join (Analysis Tools)*. ArcGIS Pro documentation, Esri. <https://doc.esri.com/en/arcgis-pro/latest/tool-reference/analysis/spatial-join.html>.
- Esri. (t.t.). *Write geoprocessing output to memory*. ArcGIS Pro documentation, Esri. <https://doc.esri.com/en/arcgis-pro/latest/help/analysis/geoprocessing/basics/the-in-memory-workspace.html>.
- QGIS Project. (t.t.). *Level up with Expressions*. QGIS Documentation, versi 3.44. https://docs.qgis.org/latest/en/docs/user_manual/expressions/index.html.
- QGIS Project. (t.t.). *PyQGIS Developer Cookbook*. QGIS Documentation, versi 3.44. https://docs.qgis.org/3.44/en/docs/pyqgis_developer_cookbook/index.html.
- QGIS Project. (t.t.). *Tasks - doing heavy work in the background*. QGIS Documentation, versi 3.44. https://docs.qgis.org/latest/en/docs/pyqgis_developer_cookbook/tasks.html.
- QGIS Project. (t.t.). *The batch processing interface*. QGIS Documentation, versi 3.44. https://docs.qgis.org/latest/en/docs/user_manual/processing/batch.html.
- QGIS Project. (t.t.). *The model designer*. QGIS Documentation, versi 3.44. https://docs.qgis.org/latest/en/docs/user_manual/processing/modeler.html.
- QGIS Project. (t.t.). *Using processing algorithms from the console*. QGIS Documentation, versi 3.44. https://docs.qgis.org/latest/en/docs/user_manual/processing/console.html.

Aplikasi Live Penulis

Berikut aplikasi web milik penulis yang bisa Anda buka langsung. Daftar ini dicek pada 7 Oktober 2026. Sumbernya bagian Aplikasi Live di <https://badarmubarok.id/#aplikasi-live>.

No.	Aplikasi	Tautan
1	SIM RHL — Sistem Informasi Manajemen Rehabilitasi Hutan dan Lahan (Fakultas Kehutanan UGM), Versi PC	https://simrhl.badarmubarok.id
2	SIM RHL, Versi HP	https://m.simrhl.badarmubarok.id
3	ScienceThink — RAD Tool (Rapid Application Development)	https://sciencethink.badarmubarok.id
4	Peta Landsystem Indonesia — peta landsystem dan peta tematik lainnya	https://landsystem.badarmubarok.id
5	Analisis Kebutuhan Unsur Hara — aplikasi Web GIS analisis unsur hara	https://unsurhara.badarmubarok.id
6	Analisa Karbon — Web GIS estimasi cadangan karbon dari citra satelit	https://karbon.badarmubarok.id
7	Aplikasi Deforestasi — deteksi perubahan tutupan hutan (2001–2025)	https://deforestasi.badarmubarok.id
8	Deteksi Area Terbakar — pemetaan tingkat keparahan area terbakar (NBR/dNBR)	https://kebakaran.badarmubarok.id
9	Mata Elang — platform deteksi konservasi berbasis AI	https://mataelang.badarmubarok.id
10	SIM Konflik — Sistem Informasi Manajemen Konflik	https://simkonflik.badarmubarok.id
11	Seri Buku GIS Kehutanan dan Pertanian — 12 buku praktik GIS (dalam bahasa Indonesia)	https://badarmubarok.id/buku-gis/

INTERMEDIATE CLASS • SERI I4

OTOMASI DASAR DENGAN PYTHON

Dari klik ke skrip: PyQGIS dan ArcPy dasar, batch, penanganan galat, model, dan jembatan menuju Masterclass.

Yang akan Anda pelajari

- 1 Python di Dalam QGIS: Bicara dengan Peta Lewat Perintah
- 2 Alat Processing dari Skrip: Satu Perintah, Satu Analisis
- 3 Skrip Batch: Satu Resep untuk Banyak Berkas
- 4 Penanganan Galat: Skrip yang Tidak Mudah Menyerah
- 5 Ekspresi: Bahasa Rumus yang Dipakai di Mana-Mana
- 6 Modeler Lanjut: Mesin dengan Tombol Pengaturan
- 7 Bab Jembatan ke Master: Proyek Mini Terpadu dan Ceklis Masuk M1



**Pindai untuk membuka
seluruh seri buku GIS**

Buku PDF, paket data, dan skrip
<https://badarmubarok.id/buku-gis/>