Lembar Contekan Python: Advanced

Bab 44-68 | rekayasa, data besar, ML dan AI, tingkat ahli

Performa dan profiling — Bab 44

```
$ python -m cProfile -o out.prof app.py
  # siapa yang lambat? [cek]
$ kernprof -l -v app.py  # profil per baris [cek]
import timeit  # ukur, jangan tebak
tm = timeit.timeit
l = tm('99 in x', 'x = list(range(100))')
s = tm('99 in x', 'x = set(range(100))')
s < l  # -> True
import tracemalloc  # pantau memori
tracemalloc.start()
x = [0] * 100_000
tracemalloc.get_traced_memory()[0] > 10**5
  # -> True
```

Pengemasan — Bab 45

```
# pyproject.toml (dari Bab 45) [ref]
[build-system]
requires = ["setuptools>=77"]
build-backend = "setuptools.build_meta"
[project]
name = "pantau-lapangan-contoh"
version = "0.1.0"
[project.scripts]
pantau-lapangan = "pantau_lapangan.cli:main"
$ python -m build  # buat sdist + wheel [cek]
$ twine check dist/*  # periksa metadata [cek]
```

Python 3.12 - 3.14 — Bab 47

```
# 3.12: generik + type
def pertama[T](xs: list[T]) -> T:
    return xs[0]
type Daftar[T] = list[T]
pertama([3, 1])  # -> 3
# 3.12: kutip sama di f-string
j = ['Jati', 'Mahoni']
f"{", ".join(j)}"  # -> 'Jati, Mahoni'
import copy, datetime  # 3.13: copy.replace
copy.replace(datetime.date(2026, 1, 1), month=6)
  # -> datetime.date(2026, 6, 1)
nama = 'Jati'  # 3.14: t-string
t = t'Pohon {nama}'
type(t).__name__, t.strings
  # -> ('Template', ('Pohon ', ''))
# 3.14: acuan maju tanpa kutip
class Simpul:
    def __init__(self, nxt: Simpul | None):
        self.nxt = nxt
Simpul(Simpul(None)).nxt.nxt is None  # -> True
# 3.14: subinterpreter [cek]
from concurrent.futures import (
    InterpreterPoolExecutor)
```

Docker dan CI (referensi) — Bab 48

```
# Dockerfile (dari Bab 48) [ref]
FROM python:3.14-slim
WORKDIR /app
USER appuser
EXPOSE 8000
$ docker build -t pantau .
  # Docker tak terpasang di sini [ref]
$ docker run -p 8000:8000 pantau
  # buka localhost:8000 [ref]
on:  # GitHub Actions (dari Bab 48) [ref]
  push:
jobs:
  uji:
    runs-on: ubuntu-latest
```

Keamanan — Bab 49

```
# token acak aman
import hashlib, hmac, secrets
len(secrets.token_hex(16))  # -> 32
hashlib.sha256(b'abc').hexdigest()[:12]
  # sidik jari data -> 'ba7816bf8f01'
# kata sandi: scrypt
k = dict(salt=b'garam', n=2**14, r=8, p=1)
h = hashlib.scrypt(b'rahasia', **k)
h2 = hashlib.scrypt(b'rahasia', **k)
hmac.compare_digest(h, h2)  # -> True
# tolak ../ (traversal)
from pathlib import Path
d = Path('data').resolve()
(d / '../rahasia.txt').resolve().is_relative_to(d)
  # -> False
import html  # cegah injeksi HTML
html.escape('<script>')  # -> '&lt;script&gt;'
$ pip-audit -r requirements.txt
  # cari paket rentan [cek]
$ bandit -r src  # cari pola berbahaya [cek]
```

Polars, DuckDB, Parquet — Bab 50

```
import polars as pl  # tabel cepat
d = pl.DataFrame(dict(k=list('aba'), v=[1, 2, 3]))
g = d.group_by('k').agg(pl.col('v').sum())
g.sort('k').to_dicts()
  # -> [{'k': 'a', 'v': 4}, {'k': 'b', 'v': 2}]
d.lazy().filter(pl.col('v') > 1).collect().shape
  # rencana malas (lazy) -> (2, 2)
# Parquet: kolom, terkompres
d.write_parquet('a.parquet')
pl.read_parquet('a.parquet').shape  # -> (3, 2)
import duckdb  # SQL pada berkas
q = "SELECT k, SUM(v) FROM 'a.parquet' GROUP BY k"
sorted(duckdb.sql(q).fetchall())
  # -> [('a', 4), ('b', 2)]
```

STAC, COG, Dask, xarray — Bab 51, 53

```
import pystac  # satu adegan = satu Item
from datetime import datetime, timezone
t = datetime(2026, 1, 1, tzinfo=timezone.utc)
i = pystac.Item('s2-1', None, None, t, {})
i.to_dict()['type']  # -> 'Feature'
from pystac_client import Client
  # cari di katalog STAC [cek]
s.read(1, window=Window(2, 2, 3, 3))
  # COG: baca sepotong (butuh jaringan) [ref]
# potongan + hitung malas
import dask.array as da
a = da.ones((1000, 1000), chunks=500)
float(a.sum().compute())  # -> 1000000.0
# larik berlabel
import numpy as np, xarray as xr
a = np.arange(6).reshape(2, 3)
x = xr.DataArray(a, dims=('t', 'k'))
x.isel(t=1).sum().item()  # -> 12
```

scikit-learn dan boosting — Bab 54

```
# data tiruan
from sklearn.datasets import make_classification
X, y = make_classification(200, random_state=0)
# validasi silang 5 lipatan
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
m = RandomForestClassifier(50, random_state=0)
s = cross_val_score(m, X, y, cv=5)
round(float(s.mean()), 2)  # -> 0.93
int(m.fit(X, y).feature_importances_.argmax())
  # fitur terpenting -> 14
import xgboost as xgb  # boosting
g = xgb.XGBClassifier(n_estimators=20)
g.fit(X, y).score(X, y) > 0.9  # -> True
```

PyTorch dan GPU — Bab 55

```
# latih: nol, hitung, mundur, langkah
import torch
x = torch.tensor([[1.], [2.], [3.]])
net = torch.nn.Linear(1, 1)
opt = torch.optim.SGD(net.parameters(), lr=.05)
for _ in range(200):
    opt.zero_grad()
    loss = ((net(x) - 2 * x) ** 2).mean()
    loss.backward()
    opt.step()
round(net.weight.item(), 1)  # -> 2.0
d = 'cuda' if torch.cuda.is_available() else 'cpu'
  # GPU bila ada
net.to(d)(x.to(d)).shape
  # model + data satu perangkat -> torch.Size([3,
  # 1])
```

Computer vision: YOLO — Bab 56

```
# IoU: tumpang tindih kotak
ar = lambda b: (b[2] - b[0]) * (b[3] - b[1])
def iou(a, b):
    w = min(a[2], b[2]) - max(a[0], b[0])
    h = min(a[3], b[3]) - max(a[1], b[1])
    i = max(w, 0) * max(h, 0)
    return i / (ar(a) + ar(b) - i)
round(iou((0, 0, 10, 10), (5, 5, 15, 15)), 3)
  # -> 0.143
from ultralytics import YOLO  # deteksi objek [cek]
m = YOLO('yolo26n.pt')  # bobot diunduh [ref]
m.train(data='data.yaml', epochs=50)
  # latih (GPU disarankan) [cek]
m.predict('foto.jpg', conf=0.25)  # deteksi [cek]
```

Optuna dan NSGA-II — Bab 57

```
import optuna  # x terbaik (benar = 2)
optuna.logging.set_verbosity(40)
def f(t):
    return (t.suggest_float('x', -5, 5) - 2) ** 2
s = optuna.samplers.TPESampler(seed=0)
st = optuna.create_study(sampler=s)
st.optimize(f, 30)
round(st.best_params['x'], 1)  # -> 2.0
# 2 tujuan = front Pareto
from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.problems import get_problem
from pymoo.optimize import minimize
p = get_problem('zdt1')
r = minimize(p, NSGA2(pop_size=20), ('n_gen', 20), s
eed=1)
r.F.shape[1]  # -> 2
```

SHAP dan MLflow — Bab 58

```
# data tiruan
from sklearn.datasets import make_classification
X, y = make_classification(200, random_state=0)
# kontribusi tiap fitur
import numpy as np, shap
from sklearn.ensemble import RandomForestClassifier
m = RandomForestClassifier(20, random_state=0)
e = shap.TreeExplainer(m.fit(X, y))
np.shape(e.shap_values(X[:5]))  # -> (5, 20, 2)
import mlflow  # catat percobaan
mlflow.set_tracking_uri('sqlite:///m.db')
with mlflow.start_run():
    mlflow.log_metric('akurasi', 0.9)
r = mlflow.search_runs(search_all_experiments=True)
len(r)  # -> 1
```

RAG, vektor, agent, MCP — Bab 60, 62

```
# tetangga terdekat
import faiss, numpy as np
v = np.random.default_rng(0).random((9, 4))
v = v.astype('float32')
ix = faiss.IndexFlatL2(4)
ix.add(v)
int(ix.search(v[:1], 2)[1][0][0])  # -> 0
import chromadb  # basis data vektor
c = chromadb.Client().create_collection('kolam')
c.add(ids=['1', '2'], documents=['padi', 'ikan'],
      embeddings=[[1, 0], [0, 1]])
q = dict(query_embeddings=[[1, .1]], n_results=1)
c.query(**q)['documents']  # -> [['padi']]
# RAG: konteks masuk prompt
konteks, tanya = 'Kolam A: pH 7.2', 'pH kolam A?'
f'Jawab dari konteks saja.\n{konteks}\n{tanya}'
  # -> 'Jawab dari konteks saja.\nKolam A: pH
  # 7.2\npH ko...
# resp = klien.generate(prompt)
  # butuh kunci [disamarkan]; tidak dijalankan [ref]
# agent: putar sampai selesai/batas
def agen(model, alat, maks=5):
    for _ in range(maks):
        a = model()
        if 'jawab' in a:
            return a['jawab']
        alat[a['alat']](a['arg'])
# model tiruan menjawab
s = iter([{'alat': 't', 'arg': 2}, {'jawab': '5'}])
agen(lambda: next(s), {'t': lambda a: a + 3})
  # -> '5'
# server + klien MCP (di memori)
import asyncio
from mcp import Client
from mcp.server import MCPServer
sv = MCPServer('kolam')
@sv.tool()
def biomassa(n: int, g: float) -> float:
    return round(n * g / 1000, 2)
async def main():
    async with Client(sv) as k:
        a = {'n': 800, 'g': 540.5}
        h = await k.call_tool('biomassa', a)
        return h.structured_content
asyncio.run(main())  # -> {'result': 432.4}
```

Deskriptor, bytecode, internals — Bab 64-65

```
# deskriptor: penjaga atribut
class Positif:
    def __set_name__(self, o, n):
        self.n = '_' + n
    def __get__(self, o, t=None):
        return getattr(o, self.n)
    def __set__(self, o, v):
        if v <= 0:
            raise ValueError('harus > 0')
        setattr(o, self.n, v)
class Pohon:
    d = Positif()
    def __init__(self, d):
        self.d = d
Pohon(32.5).d  # -> 32.5
type('B', (), {'x': 1}).x
  # kelas dibuat saat jalan -> 1
import dis  # bytecode
f = lambda a: a + 1
[i.opname for i in dis.get_instructions(f)][-2:]
  # -> ['BINARY_OP', 'RETURN_VALUE']
import sys  # hitung referensi (x + argumen)
x = []
sys.getrefcount(x)  # -> 2
```

numba, Cython, CuPy — Bab 66

```
from numba import njit  # JIT: for secepat C
@njit
def jumlah_kuadrat(n):
    s = 0
    for i in range(n):
        s += i * i
    return s
jumlah_kuadrat(1000)  # -> 332833500
$ cythonize -i ndvi_cy.pyx
  # kompilasi .pyx (butuh compiler C) [cek]
# GPU NVIDIA seperti numpy
import cupy as cp
float(cp.asarray([1., 2., 3.]).sum())  # -> 6.0
```

Celery (antrean tugas) — Bab 67

```
# mode eager: uji tanpa broker
from celery import Celery
app = Celery('t', broker='memory://',
             backend='cache+memory://')
app.conf.task_always_eager = True
@app.task
def tambah(a, b):
    return a + b
tambah.delay(2, 3).get()  # -> 5
# coba ulang otomatis
@app.task(bind=True, max_retries=3)
def rapuh(self, x):
    return x
rapuh.delay(7).get()  # -> 7
$ celery -A tugas worker --concurrency=3
  # tiga proses worker [cek]
redis.Redis.from_url(url, protocol=2)
  # Redis 5 (butuh server) [ref]
```

Buku "Python dari Nol sampai Sangat Advanced", Lampiran B. Python 3.14. Setiap baris kode dijalankan sungguhan; "->" = hasil asli. Angka numpy ditulis tanpa pembungkus np.float64(...).
[cek] = opsi atau fungsi diperiksa ada (tidak dijalankan penuh). [ref] = referensi, tidak dijalankan di sini. $ = perintah terminal.