

Python Cheat Sheet: Intermediate

Ch. 16-43 | tidy code, data, statistics, first applications

OOP: classes and inheritance

Ch. 16

```
class Tree: # blueprint
    def __init__(self, kind, d):
        self.kind, self.d = kind, d
    def __str__(self):
        return f'{self.kind} {self.d}'
str(Tree('Teak', 32.5))
# make an object -> 'Teak 32.5'
class Big(Tree): # inheritance
    def __str__(self):
        return 'BIG ' + super().__str__()
str(Big('Teak', 45)) # -> 'BIG Teak 45'
class Pond: # property: no ()
    @property
    def ph(self):
        return 7.0
Pond().ph # -> 7.0
# dataclass
from dataclasses import dataclass
@dataclass
class Fish:
    name: str
Fish('catfish')
# automatic __repr__ -> Fish(name='catfish')
```

Environments, pytest, ruff, logging

Ch. 17-19

```
$ python -m venv .venv
# your own Python room [chk]
$ .venv\Scripts\activate
# Linux: source .venv/bin/activate [ref]
$ uv add pandas # add a dependency [chk]
$ uv run pytest # run in the project env [chk]
$ uv sync # match uv.lock [chk]
$ pytest -x -k name # stop at first failure [chk]
import pytest # test_*.py
P = [(1, 1), (2, 4)]
@pytest.mark.parametrize('n,h', P)
def test_k(n, h):
    assert n * n == h
def test_e():
    with pytest.raises(ZeroDivisionError):
        1 / 0 # -> 3 passed
$ ruff check --fix # lint + auto-fix [chk]
$ ruff format . # format code [chk]
import logging, sys # levelled log
logging.basicConfig(level=20, force=True,
                    stream=sys.stdout)
logging.info('start') # -> INFO:root:start
```

Git and GitHub

Ch. 20

```
$ git init -b main # start a repo (a.txt exists)
$ git add -A # stage changes
$ git commit -m 'message' # save a snapshot
$ git switch -c feature # new branch
$ git switch main # go back
$ git merge feature # merge a branch
$ git log --oneline # short history
$ git push -u origin main # send to GitHub [chk]
# .gitignore: .venv/ .env (secrets!)
```

APIs with requests

Ch. 21

```
import requests
# url = API address (here: a local server)
# always set a timeout
r = requests.get(url, params={'q': 1},
                 timeout=10)
r.status_code, r.json() # -> (200, {'n': 3})
# missing = 404
r2 = requests.get(url + '/x', timeout=10)
r2.status_code # -> 404
r.raise_for_status() # error on 4xx/5xx
```

Regex

Ch. 22

```
import re
t = 'JT-0042, MR-0107, xx-12'
re.findall(r'\b[A-Z]{2}-\d{4}\b', t)
# all codes -> ['JT-0042', 'MR-0107']
re.search(r'd=(\d+)', 'd=45 cm')[1]
# capture a group () -> '45'
re.sub(r's+', ' ', 'a b c')
# replace -> 'a b c'
# named groups
m = re.search(r'(?P<k>\w+)=(?P<n>\d+)', 't=14')
m.groupdict() # -> {'k': 't', 'n': '14'}
re.fullmatch(r'\d{3}', '123') is not None
# whole match? -> True
```

numpy and pandas

Ch. 23

```
import numpy as np
a = np.array([480, 500, 510])
a[a > 490] # boolean mask -> array([500, 510])
import pandas as pd # table
df = pd.DataFrame({'k': list('ABC'),
                  'j': ['catfish', 'tilapia', 'catfish'],
                  'n': [800, 1250, 600]})
df.shape # -> (3, 3)
df['kg'] = df.n * 0.5 # new column
df[df.kg > 300].k.tolist()
# filter rows -> ['A', 'B']
df.groupby('j')['kg'].sum().to_dict()
# group summary -> {'catfish': 700.0, 'tilapia': 625.0}
df.sort_values('kg').k.iloc[-1]
# the largest -> 'B'
```

numpy and pandas ...

Ch. 23

```
# join tables
h = pd.DataFrame({'j': ['catfish', 'tilapia'],
                  'rp': [3, 2]})
df.merge(h, on='j').shape # -> (3, 5)
df.to_csv('a.csv', index=False) # save CSV
pd.read_csv('a.csv').shape # read CSV -> (3, 4)
```

SQL: sqlite3 and SQLAlchemy

Ch. 24

```
import sqlite3 # ? stops injection
c = sqlite3.connect(':memory:')
c.execute('CREATE TABLE p (j TEXT, d REAL)')
r = [(('Teak', 32.5), ('Mahogany', 28.0))]
c.executemany('INSERT INTO p VALUES (?,?)', r)
# parameterised query
q = 'SELECT j FROM p WHERE d > ?'
c.execute(q, (30,)).fetchall() # -> [(('Teak',)]
# ORM: class = table
import sqlalchemy as sa
from sqlalchemy.orm import (DeclarativeBase,
                             Mapped, Session, mapped_column)
class B(DeclarativeBase):
    pass
class Sector(B):
    tablename__ = 's'
    id: Mapped[int] = mapped_column(
        primary_key=True)
    name: Mapped[str]
e = sa.create_engine('sqlite:///')
B.metadata.create_all(e)
with Session(e) as s:
    s.add(Sector(name='Fish'))
    s.commit()
    n = len(s.scalars(sa.select(Sector)).all())
n # -> 1
with e.connect() as k: # raw SQL
    q = sa.text('SELECT COUNT(*) FROM s')
    cnt = k.execute(q).scalar()
cnt # -> 1
```

Statistics and statsmodels

Ch. 26-27

```
from scipy import stats # two-group t-test
a, b = [30, 32, 31, 29], [35, 36, 34, 37]
stats.ttest_ind(a, b).pvalue < 0.05 # -> True
round(stats.norm.cdf(30, 25, 5), 3)
# normal probability -> 0.841
round(stats.pearsonr(a, b)[0], 2)
# correlation -> -0.4
# OLS: [intercept, slope]
import statsmodels.api as sm
X = sm.add_constant([1, 2, 3, 4, 5])
y = [2.1, 3.9, 6.2, 7.8, 10.1]
res = sm.OLS(y, X).fit()
res.params.round(2).tolist() # -> [0.05, 1.99]
round(res.rsquared, 3) # R squared -> 0.997
# logistic: odds ratio
import numpy as np, pandas as pd
import statsmodels.formula.api as smf
d = pd.DataFrame({'x': range(1, 9),
                  'y': [0, 0, 1, 0, 1, 0, 1, 1]})
r = smf.logit('y ~ x', d).fit(dis=0)
round(float(np.exp(r.params['x'])), 2) # -> 1.81
```

Generators, decorators, closures

Ch. 30-31

```
def evens(n): # yield = one at a time
    for i in range(2, n + 1, 2):
        yield i
list(evens(10)) # -> [2, 4, 6, 8, 10]
# generator expression
g = (i * i for i in range(3))
next(g), next(g) # -> (0, 1)
def multiplier(k): # closure remembers k
    return lambda x: x * k
multiplier(3)(7) # -> 21
import functools # decorator + wraps
def count_calls(f):
    @functools.wraps(f)
    def b(*a):
        b.n += 1
        return f(*a)
        b.n = 0
    return b
@count_calls
def add(a, b):
    return a + b
add(1, 2), add.n # n = number of calls -> (3, 1)
```

Type hints and Pydantic

Ch. 33-34

```
# int or None
def find(xs: list[str], x: str) -> int | None:
    return xs.index(x) if x in xs else None
find(['a', 'b'], 'b'), find(['a'], 'z')
# -> (1, None)
$ mypy --strict app.py
# check types before running [chk]
# model = gatekeeper
from pydantic import BaseModel, Field
class Tree(BaseModel):
    kind: str
    d: float = Field(gt=0)
Tree(kind='Teak', d=32.5)
# '32.5' becomes a number -> Tree(kind='Teak',
# d=32.5)
try: # tidy error
    Tree(kind='Teak', d=-5)
except Exception as e:
```

Type hints and Pydantic ...

Ch. 33-34

```
message = e.errors()[0]['msg']
message # -> 'Input should be greater than 0'
# settings (can come from env)
from pydantic.settings import BaseSettings
class Config(BaseSettings):
    folder: str = 'result'
Config().folder # -> 'result'
```

Async and threads

Ch. 35

```
import asyncio # concurrently
async def f(n):
    await asyncio.sleep(0.01)
    return n * 2
async def main():
    return await asyncio.gather(f(1), f(2))
asyncio.run(main()) # -> [2, 4]
async def g(): # TaskGroup
    async with asyncio.TaskGroup() as tg:
        t = tg.create_task(f(1))
        return t.result()
asyncio.run(g()) # -> 2
# threads for I/O waits
import concurrent.futures as cf
with cf.ThreadPoolExecutor(2) as p:
    result = list(p.map(abs, [-1, -2]))
result # -> [1, 2]
```

Web: FastAPI and Flask

Ch. 36

```
# path + parameter
from fastapi import FastAPI
from fastapi.testclient import TestClient
app = FastAPI()
@app.get('/pond/{n}')
def pond(n: str, ph: float = 7.0):
    return {'n': n, 'ph': ph}
c = TestClient(app)
c.get('/pond/A?ph=6.8').json()
# -> {'n': 'A', 'ph': 6.8}
c.get('/pond/A?ph=x').status_code
# wrong type = 422 -> 422
$ uvicorn app:app --reload
# start the server [chk]
# Flask: the small one
from flask import Flask
w = Flask(__name__)
@w.get('/hello')
def hello():
    return {'p': 'hi'}
w.test_client().get('/hello').json
# -> {'p': 'hi'}
```

GeoPandas and Rasterio

Ch. 38-39

```
import geopandas as gpd # points + CRS
from shapely.geometry import Point
pts = [Point(110, -7), Point(110.01, -7.01)]
g = gpd.GeoDataFrame({'n': [1, 2]}, geometry=pts,
                    crs='EPSG:4326')
g.crs.to_epsg() # -> 4326
u = g.to_crs(32749) # to metres (UTM 49S)
round(u.geometry[0].distance(u.geometry[1]))
# distance (m) -> 1563
round(u.buffer(100).area[0])
# buffer area (m2) -> 31365
g.to_file('a.gpkg') # save/read GeoPackage
gpd.read_file('a.gpkg').shape # -> (2, 2)
# write a GeoTIFF
import numpy as np, rasterio
from rasterio.transform import from_origin
a = np.arange(4, dtype='float32').reshape(2, 2)
t = from_origin(110, -7, 0.1, 0.1)
with rasterio.open('a.tif', 'w', driver='GTiff',
                  height=2, width=2, count=1,
                  dtype='float32', crs='EPSG:4326',
                  transform=t) as d:
    d.write(a, 1)
# read band 1 + CRS
with rasterio.open('a.tif') as s:
    b, e = s.read(1), s.crs.to_epsg()
(b.mean().item(), e) # -> (1.5, 4326)
# NDVI
n, r = np.array([.8, .6]), np.array([.2, .3])
np.round((n - r) / (n + r), 2).tolist()
# -> [0.6, 0.33]
```

Context managers and automation

Ch. 32, 41-43

```
# ignore one error type
from contextlib import suppress
with suppress(FileNotFoundError):
    open('missing').close()
# run another program
import subprocess, sys
cmd = [sys.executable, '-c', 'print(2)']
p = subprocess.run(cmd, text=True,
                  capture_output=True)
p.stdout.strip() # -> '2'
import argparse # command-line arguments
ap = argparse.ArgumentParser()
ap.add_argument('--n', type=int, default=3)
ap.parse_args(['--n', '5']).n # -> 5
$ streamlit run app.py
# dashboard in the browser [chk]
```

Book "Python from Zero to Very Advanced", Appendix B. Python 3.14. Every code line was really executed; "->" = real result. numpy numbers are shown without the np.float64(...) wrapper.

[chk] = option or function checked to exist (not fully run). [ref] = reference, not run here. \$ = terminal command.