

Python Cheat Sheet: Basic

Ch. 1-15 | from zero to useful small programs

Terminal and Python

Ch. 3-4

```
$ cd project_folder # change folder [ref]
$ python hello.py # run a script [ref]
$ python --version # check version
print('Hello', 3, sep=', ', end='')
# print -> Hello, 3!
# comment: a note for humans
```

Variables and data types

Ch. 5

```
n, h, ok = 42, 1.5, True
# int, float, bool
name = 'Teak' # str (text)
type(h).__name__ # type name -> 'float'
int('42') + 1 # text to number -> 43
float('2.5') # text to decimal -> 2.5
str(7) + ' kg' # number to text -> '7 kg'
bool(0), bool('a')
# empty/zero = False -> (False, True)
f'{3.14159:.2f}' # two decimals -> '3.14'
f'{1234567:,}'
# thousands separator -> '1,234,567'
x = None # no value
```

Input

Ch. 5-6

```
age = int(input('Age? '))
# input() is always text!
age + 1 # after converting to int -> 43
```

Operators

Ch. 6

```
7 / 2, 7 // 2, 7 % 2
# divide, floor, rest -> (3.5, 3, 1)
2 ** 10 # power -> 1024
divmod(1000, 30)
# (quotient, rest) -> (33, 10)
round(3.14159, 2) # rounding -> 3.14
1 < 2 < 3 # chained comparison -> True
(1 > 2) or not (3 > 4)
# and, or, not -> True
'ea' in 'Teak' # contained in? -> True
n = 5 # add and store
n += 2
```

Conditions

Ch. 7

```
n = 520 # if / elif / else
if n >= 500:
    s = 'ready'
elif n >= 400:
    s = 'almost'
else:
    s = 'not yet'
s # -> 'ready'
'ready' if n >= 500 else 'not yet'
# one-liner -> 'ready'
kind = 'cattle' # pattern matching
match kind:
    case 'cattle' | 'goat':
        k = 'ruminant'
    case _:
        k = 'other'
k # -> 'ruminant'
```

Loops

Ch. 8

```
total = 0 # accumulator
for b in [3, 5, 7]:
    total += b
total # -> 15
list(range(0, 10, 3))
# start, stop (excluded), step -> [0, 3, 6, 9]
list(enumerate('ab', 1))
# index + item -> [(1, 'a'), (2, 'b')]
g = 1800 # while + break
while True:
    g += 150
    if g >= 2400:
        break
g # -> 2400
[i for i in range(5) if i % 2 == 0]
# list comprehension -> [0, 2, 4]
```

Lists and slicing

Ch. 9

```
xs = [3, 1, 2] # list
xs.append(4) # add at the end
xs[0], xs[-1] # first, last -> (3, 4)
xs[1:3] # slice [start:stop) -> [1, 2]
sorted(xs)
# sorted (original kept) -> [1, 2, 3, 4]
xs.sort(reverse=True) # sort in place
```

Lists and slicing ...

Ch. 9

```
xs[::-1] # reversed -> [1, 2, 3, 4]
len(xs), sum(xs), max(xs)
# count, sum, largest -> (4, 10, 4)
```

Dict, tuple, set

Ch. 9

```
d = {'Teak': 32.5, 'Mahogany': 28.0}
# dict: key -> value
d['Meranti'] = 45.2 # add/change
d.get('Sengon', 0)
# default if missing -> 0
{k: v for k, v in d.items() if v > 30}
# dict comprehension -> {'Teak': 32.5, 'Meranti': 45.2}
latitude, longitude = (-6.6, 106.8)
# tuple: immutable
{1, 2, 3} & {2, 3, 4}
# set: intersection -> {2, 3}
{1, 2} | {2, 3} # set: union -> {1, 2, 3}
```

Functions

Ch. 10

```
def area(r, pi=3.14): # default value pi
    return pi * r * r
area(2) # uses the default -> 12.56
area(r=2, pi=3) # named arguments -> 12
times = lambda a, b: a * b # tiny function
times(3, 4) # -> 12
sorted(['bb', 'a'], key=len)
# sort key -> ['a', 'bb']
# no return means the result is None
```

Modules

Ch. 11

```
import math # import a module
round(math.pi, 3) # -> 3.142
from datetime import date, timedelta
# import parts
date(2026, 1, 15) + timedelta(days=270)
# date arithmetic -> datetime.date(2026, 10, 12)
date(2026, 10, 12).strftime('%d-%m-%Y')
# format a date -> '12-10-2026'
import random # seeded random
random.seed(7)
random.sample(range(1, 13), 3)
# -> [6, 3, 7]
```

Errors

Ch. 12

```
def to_number(t): # catch the error
    try:
        return float(t)
    except ValueError:
        return None
to_number('12.5'), to_number('abc')
# safe from bad text -> (12.5, None)
try: # raise, except, finally
    raise ValueError('broken')
except (ValueError, KeyError) as e:
    message = str(e)
finally:
    pass
message # -> 'broken'
# your own error
class DataError(Exception):
    pass
assert 1 + 1 == 2, 'must be 2'
# check an assumption
import logging # log a warning
logging.warning('pH odd')
```

Files and Path

Ch. 13

```
from pathlib import Path # file path
p = Path('a.txt')
# w overwrite, a append, r read
with open(p, 'w', encoding='utf-8') as f:
    f.write('1\n2\n')
# read line by line
with open(p, encoding='utf-8') as f:
    total = sum(int(b) for b in f)
total # -> 3
p.read_text(encoding='utf-8')
# read all -> '1\n2\n'
q = Path('data/b.txt') # parts of a path
q.name, q.suffix, q.stem
# -> ('b.txt', '.txt', 'b')
p.exists() # does it exist? -> True
```

CSV and JSON

Ch. 14

```
import csv, json # built-in modules
# newline='' required
with open('t.csv', 'w', newline='', encoding='utf-8') as f:
    w = csv.DictWriter(f, ['k', 'n'])
    w.writeheader()
    w.writerow({'k': 'A', 'n': 800})
# all values become text!
with open('t.csv', newline='', encoding='utf-8') as f:
    rows = list(csv.DictReader(f))
rows # -> [{'k': 'A', 'n': '800'}]
json.dumps({'a': '0'}, ensure_ascii=False)
# object to JSON text -> '{"a": "0"}'
json.loads({'a': [1, 2]})['a']
# JSON text to object -> [1, 2]
# NEVER pickle.load from a stranger!
```

Excel and PDF

Ch. 15

```
import openpyxl as xl # Excel library
wb = xl.Workbook() # write .xlsx
ws = wb.active
ws.append(['pond', 'kg'])
ws.append(['A', 432.4])
wb.save('a.xlsx')
# read a cell
wb2 = xl.load_workbook('a.xlsx')
wb2.active['B2'].value # -> 432.4
# write a PDF
from reportlab.pdfgen import canvas
c = canvas.Canvas('a.pdf')
c.drawString(72, 750, 'Report')
c.save()
from pypdf import PdfReader # read a PDF
len(PdfReader('a.pdf').pages) # -> 1
```