

VARIABLES & TYPES

x = 5; name = "Ann"	assign (no declaration)
int float str bool	5 · 3.14 · "hi" · True/False
list tuple dict set	[1] · {1,} · {"a":1} · {1}
None	"no value"; test with <code>is None</code>
type(x) isinstance(x, int)	inspect type
int("5") float("2.5") str(5)	convert types
a, b = b, a	swap / unpack

NUMBERS & OPERATORS

+ - * /	/ always returns float: 7/2 = 3.5
7 // 2	floor division = 3 (-7//2 = -4)
7 % 3	remainder = 1
2 ** 10	power = 1024
divmod(7, 2)	(3, 1)
round(3.14159, 2)	3.14 (abs, pow also built in)
x += 1	also -= *= /=
== != < > <= >=	compare (chain: 1 < x < 5)
and or not in is	logic / membership / identity

STRINGS

f"{name} {pi:.2f}"	f-string, 2 decimals
f"{n:}, {r:.1%} {s:>8}"	1,234 · 25.6% · right-align
s.strip() .lower() .upper()	trim / change case
s.split(",") ", ".join(lst)	to list / from list
s.replace("a", "b")	replace all
s.startswith("a") .endswith()	prefix / suffix test
s.find("x")	index, or -1 if missing
s.isdigit() .title() len(s)	checks, case, length
"ab" in s	substring test

Strings are immutable: methods return a new string.

INDEXING & SLICING

t = "abcdef"	# [start:stop:step]
t[0] # 'a'	t[-1] # 'f'
t[1:4] # 'bcd'	(stop excluded)
t[:2] # 'ab'	t[-2:] # 'ef'
t[::2] # 'ace'	t[::-1] # 'fedcba'

Works the same on lists and tuples. Index starts at 0.

LISTS

L = [3, 1, 2]	ordered, mutable
L.append(4) L.extend([5, 6])	add one / many at end
L.insert(0, 9)	insert at index
L.remove(9) L.pop() L.pop(0)	by value / last / by index
L.sort() L.reverse()	in place, returns None
sorted(L)	new sorted list
L.index(2) L.count(1)	position / occurrences
L + M L * 2 2 in L	concat / repeat / membership

TUPLES, DICTS & SETS

t = (1, 2); a, b = t	immutable, unpackable
d = {"a": 1}; d["b"] = 2	add / update key
d["a"] d.get("z", 0)	KeyError / safe default
d.keys() .values() .items()	views to loop over
d.pop("a") del d["b"]	remove key
d.update(e) d.setdefault(k, v)	merge / set if missing
for k, v in d.items():	loop key and value
s = {1, 2}; s.add(3)	unique, unordered
s t s & t s - t	union / intersect / difference
set(lst)	remove duplicates

COMPREHENSIONS

```
[x*x for x in range(5) if x % 2 == 0]
# [0, 4, 16]
{w: len(w) for w in ["a", "bb"]}
# {'a': 1, 'bb': 2}
{c for c in "aab"} # {'a', 'b'}
[a+b for a in "xy" for b in "12"]
```

Pattern: `[expression for item in iterable if condition]`

IF / ELIF / ELSE

```
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"
label = "pass" if score >= 50 else "fail"
```

Falsy: 0 "" [] {} None False. Indentation (4 spaces) defines blocks.

LOOPS

```
for i in range(3): # 0 1 2
    print(i)
for i in range(10, 0, -3): # 10 7 4 1
    ...
for i, v in enumerate(items, start=1):
    print(i, v)
for a, b in zip(names, ages):
    print(a, b)
while n > 0:
    n -= 1
    if n == 5: continue # skip rest
    if n == 2: break # leave loop
```

FUNCTIONS

```
def total(a, b=2, *args, **kw) -> int:
    """Docstring."""
    return a + b + sum(args) + sum(kw.values())

total(1) # 3
total(1, 2, 3, 4, z=5) # 15
sq = lambda x: x * 2 # small anonymous fn
# hints: x: int, name: str = "n", -> str

*args = extra positional (tuple); **kw = extra keyword (dict).
No return means None is returned.
```

ERRORS & FILES

```
try:
    n = int(text)
except ValueError as e:
    print("bad number:", e)
else:
    print("ok") # no error raised
finally:
    print("always") # cleanup
    raise ValueError("msg")

with open("f.txt", encoding="utf-8") as fh:
    for line in fh:
        print(line.rstrip())
with open("o.txt", "w") as fh: # "a" = append
    fh.write("hi\n")
```

Common errors: `KeyError`, `IndexError`, `TypeError`, `FileNotFoundError`.

MODULES & BUILT-INS

```
import math
import pandas as pd
from collections import Counter
from pathlib import Path
```

len sum min max abs	size / total / extremes
sorted(xs, key=len, reverse=True)	sort by function
sorted(d.items(), key=lambda kv: kv[1])	sort dict by value
any(xs) all(xs)	at least one / every truthy
list(map(str, [1, 2]))	apply to each: ['1','2']
list(filter(lambda n: n > 1, xs))	keep matches
range zip enumerate reversed	iteration helpers
Counter("aab").most_common(1)	[('a', 2)]

CLASSES & DATACLASS

```
from dataclasses import dataclass

@dataclass
class Person:
    name: str
    age: int = 0
    def greet(self):
        return f"Hi {self.name}"

p = Person("Ann", 30)
print(p) # Person(name='Ann', age=30)
p.greet() # 'Hi Ann'
```

A dataclass auto-writes `__init__`, `__repr__` and `==`. Plain class: define `__init__(self, ...)` yourself.

COMMON GOTCHAS

Mutable default: `def f(x, l=[])` shares one list across calls. Use `l=None`, then `if l is None: l = []`.

= vs ==: `=` assigns, `==` compares. Use `is` only for None.

Integer division: `/` gives float, `//` floors (toward minus infinity).

Copying lists: `b = a` is the same list. Use `a.copy()` or `a[:]`; nested data needs `copy.deepcopy(a)`.

Floats: `0.1 + 0.2 == 0.3` is False. Use `math.isclose`.

Modifying a list while looping over it skips items. Loop over a copy or use a comprehension.