The Python Cheat Sheet

The Python Cheat Sheet · WebsiteSetup.org

What is Python

Python is a readable, general-purpose language for web backends, data, automation, and scripting. The current stable release is 3.13, and 3.14 is the newest in 2026. Run it with `python3`. Indentation of four spaces defines code blocks, not braces.

Variables and data types

Python uses dynamic typing. Assign a value and the type is inferred.

Example	Type	Meaning
`x = 5`	int	Whole number
`pi = 3.14`	float	Decimal number
`name = "Ada"`	str	Text
`ok = True`	bool	True or False
`z = None`	NoneType	No value

Check with `type(x)`. Build text with f-strings: `f"Hi {name}, x={x}"`.

Strings

String methods return new strings and never change the original.

Operation	Result
`len(s)`	Character count
`s[1:4]`	Slice from index 1 up to 4
`s.upper()` / `s.lower()`	Change case
`s.strip()`	Trim whitespace
`s.split(",")`	Split into a list
`",".join(list)`	Join a list into text
`s.replace("a", "b")`	Swap substrings
`"b" in s`	Membership test

Numbers and math

Single-slash division always returns a float, so `7 / 2 == 3.5`.

Operator	Meaning	Example
`+ - * /`	Basic arithmetic	`7 / 2 == 3.5`
`//`	Floor division	`7 // 2 == 3`
`%`	Remainder	`7 % 2 == 1`
`**`	Power	`2 ** 3 == 8`

Also: `int()`, `float()`, `round(x, 2)`, `abs()`, and `import math` for `math.sqrt()` and `math.pi`.

Lists (mutable)

Lists hold an ordered, changeable sequence.

```
xs = [1, 2, 3]
xs.append(4)        # add to end
xs.insert(0, 9)     # add at index 0
xs.remove(2)        # remove first match
last = xs.pop()     # remove and return last
part = xs[1:3]      # slice
xs.sort()           # in place, returns None
new = sorted(xs)    # returns a new list
doubled = [x * 2 for x in xs if x > 1]
```

Tuples (immutable)

Fixed groups of values that cannot change.

```
t = (1, 2)
a, b = t        # unpacking
```

Dictionaries

Key-to-value maps with fast lookups by name.

```
d = {"a": 1, "b": 2}
d["a"]                  # 1
d.get("z", 0)           # safe default
d.keys(); d.values(); d.items()
"a" in d                # membership
sq = {k: v * 2 for k, v in d.items()}
```

Sets

Unordered collections of unique values.

```
s = {1, 2, 3}
s.add(4)
a = {1, 2, 3}; b = {3, 4, 5}
a | b      # union
a & b      # intersection
a - b      # difference
```

Conditionals

Branch on truth. Four-space indentation groups each block.

```
if x > 10:
    print("big")
elif x > 0:
    print("small")
else:
    print("zero or less")

y = "hi" if x else "lo"    # ternary
```

Pattern matching (Python 3.10 and later):

```
match command:
    case "start":
        print("starting")
    case _:
        print("unknown")
```

Loops

Walk sequences with `for`; helpers cover the common cases.

```
for i in range(3):          # 0, 1, 2
    print(i)

for i, v in enumerate(xs):  # index and value
    print(i, v)

for a, b in zip(xs, ys):    # pair two lists
    print(a, b)

while count < 5:
    count += 1
    if count == 3:
        continue
    if count == 4:
        break
```

Functions

Reusable logic with defaults, variadic args, and type hints.

```
def greet(name: str, loud: bool = False) -> str:
    text = f"Hi {name}"
    return text.upper() if loud else text

def total(*args, **kwargs):
    return sum(args)

sq = lambda x: x * x
```

Comprehensions

Build a new collection in one line.

```
squares = [x * x for x in range(5)]          # list
lookup = {x: x * x for x in range(5)}        # dict
unique = {x % 3 for x in range(10)}          # set
```

Files

The `with` block closes files automatically.

```
with open("f.txt", "r") as f:
    text = f.read()
    lines = f.readlines()

with open("f.txt", "w") as f:
    f.write("hello\n")
```

Modes: `r` read, `w` overwrite, `a` append.

Exceptions

Handle errors instead of crashing.

```
try:
    value = int("abc")
except ValueError as err:
    print("bad number:", err)
else:
    print("all good:", value)
finally:
    print("done")

raise ValueError("bad")
```

Common: `ValueError`, `KeyError`, `IndexError`, `TypeError`, `FileNotFoundError`, `ZeroDivisionError`.

Modules and imports

```
import math
from math import pi

if __name__ == "__main__":
    print(pi)
```

Classes (OOP basics)

```
class Dog:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return f"{self.name} says woof"

class Puppy(Dog):
    def speak(self):
        return f"{self.name} says yip"
```

Handy built-ins and pip/venv

Function	What it does
`print()`	Write output
`input()`	Read a line from the user
`len()`	Count items
`range()`	Generate numbers
`enumerate()`	Loop with an index
`zip()`	Pair sequences
`map()` / `filter()`	Transform or select items

```
pip install requests
python3 -m venv .venv
source .venv/bin/activate
```

Mini-recipes

```
# Read a file line by line
with open("f.txt") as f:
    for line in f:
        print(line.strip())

# Filtered comprehension: even numbers
evens = [n for n in range(20) if n % 2 == 0]

# Dict from two lists
d = dict(zip(["a", "b"], [1, 2]))

# A typed function
def add(a: int, b: int) -> int:
    return a + b

# Safe integer input
try:
    age = int(input("Age: "))
except ValueError:
    age = 0
```

Common gotchas

- Indentation defines blocks. Four spaces; never mix tabs and spaces, or you get an `IndentationError`.
- `/` is always float division. Use `//` for a floor.
- Mutable default args are shared. Use `def f(x=None)`, then `x = x or []`.
- `==` compares values; `is` compares identity. Use `is` only for `None`.
- f-strings need 3.6 or later; `match/case` needs 3.10 or later.
- Lists are mutable, tuples are not.
- Use a venv per project. Python 2 is dead; use `python3` on 3.11 or newer.