Python Cheatsheet

By Dejan Panovski • Updated on Mar 14, 2026 • Download PDF

Quick reference for Python 3 syntax, data types, string and list methods, control flow, functions, file I/O, and exception handling.

Python is a general-purpose programming language known for its readable syntax. This cheatsheet covers Python 3 essentials: data types, operators, string and list methods, control flow, functions, file I/O, exception handling, and comprehensions.

Data Types

Core Python types and how to inspect them.

`42, 3.14, -7`	int, float literals
`True, False`	bool
`"hello", 'world'`	str (immutable)
`[1, 2, 3]`	list — mutable, ordered
`(1, 2, 3)`	tuple — immutable, ordered
`{"a": 1, "b": 2}`	dict — key-value pairs
`{1, 2, 3}`	set — unique, unordered
`None`	Absence of a value
`type(x)`	Get type of X
`isinstance(x, int)`	Check if X is of a given type

Operators

Arithmetic, comparison, logical, and membership operators.

`+, -, *, /`	Add, subtract, multiply, divide
`//`	Floor division
`%`	Modulo (remainder)
`**`	Exponentiation
`==, !=`	Equal, not equal
`<, >, <=, >=`	Comparison
`and, or, not`	Logical operators
`in, not in`	Membership test
`is, is not`	Identity test
`x if cond else y`	Ternary (conditional) expression

String Methods

Common operations on `str` objects. Strings are immutable — methods return a new string.

`s.upper() / s.lower()`	Convert to upper / lowercase
`s.strip()`	Remove leading and trailing whitespace
`s.lstrip() / s.rstrip()`	Remove leading / trailing whitespace
`s.split(",")`	Split on delimiter, return list
`",".join(lst)`	Join list elements into a string
`s.replace("old", "new")`	Replace all occurrences
`s.find("x")`	First index of substring (-1 if not found)
`s.startswith("x")`	`True` if string starts with prefix
`s.endswith("x")`	`True` if string ends with suffix
`s.count("x")`	Count non-overlapping occurrences
`s.isdigit() / s.isalpha()`	Check all digits / all letters
`len(s)`	String length
`s[i], s[i:j], s[::step]`	Index, slice, step
`"x" in s`	Substring membership test

String Formatting

Embed values in strings.

`f"Hello, {name}"`	f-string (Python 3.6+)
`f"{val:.2f}"`	Float with 2 decimal places
`f"{val:>10}"`	Right-align in 10 characters
`f"{val:,}"`	Thousands separator
`f"{{name}}"`	Literal braces in output
`"Hello, {}".format(name)`	str.format()
`"Hello, %s" % name`	%-formatting (legacy)

List Methods

Common operations on `list` objects.

`lst.append(x)`	Add element to end
`lst.extend([x, y])`	Add multiple elements to end
`lst.insert(i, x)`	Insert element at index `i`
`lst.remove(x)`	Remove first occurrence of `x`
`lst.pop()`	Remove and return last element
`lst.pop(i)`	Remove and return element at index `i`
`lst.index(x)`	First index of `x`
`lst.count(x)`	Count occurrences of `x`
`lst.sort()`	Sort in place (ascending)
`lst.sort(reverse=True)`	Sort in place (descending)
`lst.reverse()`	Reverse in place
`lst.copy()`	Shallow copy
`lst.clear()`	Remove all elements
`len(lst)`	List length
`lst[i], lst[i:j]`	Index and slice
`x in lst`	Membership test

Dictionary Methods

Common operations on `dict` objects.

`d[key]`	Get value by key (raises `KeyError` if missing)
`d.get(key)`	Get value or `None` if key not found
`d.get(key, default)`	Get value or default if key not found
`d[key] = val`	Set or update a value
`del d[key]`	Delete a key
`d.pop(key)`	Remove key and return its value
`d.keys()`	View of all keys
`d.values()`	View of all values
`d.items()`	View of all (key, value) pairs
`d.update(d2)`	Merge another dict into `d`
`d.setdefault(key, val)`	Set key if missing, return value
`key in d`	Check if key exists
`len(d)`	Number of key-value pairs
`{**d1, **d2}`	Merge dicts (Python 3.9+: `d1 | d2`)

Sets

Common operations on `set` objects.

`set()`	Empty set (use this, not `{}`)
`s.add(x)`	Add element
`s.remove(x)`	Remove element (raises `KeyError` if missing)
`s.discard(x)`	Remove element (no error if missing)
`s1 | s2`	Union
`s1 & s2`	Intersection
`s1 - s2`	Difference
`s1 ^ s2`	Symmetric difference
`s1 <= s2`	Subset check
`x in s`	Membership test

Control Flow

Conditionals, loops, and flow control keywords.

`if cond: / elif cond: / else:`	Conditional blocks
`for item in iterable:`	Iterate over a sequence
`for i, v in enumerate(lst):`	Iterate with index and value
`while cond:`	Loop while condition is true
`break`	Exit loop immediately
`continue`	Skip to next iteration
`pass`	No-op placeholder
`match x: / case val:`	Pattern matching (Python 3.10+)

Functions

Define and call functions.

`def func(x, y):`	Define a function
`return val`	Return a value
`def func(x=0):`	Default argument
`def func(*args):`	Variable positional arguments
`def func(**kwargs):`	Variable keyword arguments
`lambda x: x * 2`	Anonymous (lambda) function
`func(y=1, x=2)`	Call with keyword arguments
`result = func(x)`	Call and capture return value

File I/O

Open, read, and write files.

`open("f.txt", "r")`	Open for reading (default mode)
`open("f.txt", "w")`	Open for writing — creates or overwrites
`open("f.txt", "a")`	Open for appending
`open("f.txt", "rb")`	Open for reading in binary mode
`with open("f.txt") as f:`	Open with context manager (auto-close)
`f.read()`	Read entire file as a string
`f.read(n)`	Read n characters
`f.readline()`	Read one line
`f.readlines()`	Read all lines into a list
`for line in f:`	Iterate over lines
`f.write("text")`	Write string to file
`f.writelines(lst)`	Write list of strings

Exception Handling

Catch and handle runtime errors.

`try:`	Start guarded block
`except ValueError:`	Catch a specific exception
`except (TypeError, ValueError):`	Catch multiple exception types
`except Exception as e:`	Catch and bind exception to variable
`else:`	Run if no exception was raised
`finally:`	Always run — use for cleanup
`raise ValueError("msg")`	Raise an exception
`raise`	Re-raise the current exception

Comprehensions

Build lists, dicts, sets, and generators in one expression.

`[x for x in lst]`	List comprehension
`[x for x in lst if cond]`	Filtered list comprehension
`[f(x) for x in range(n)]`	Comprehension with expression
`{k: v for k, v in d.items()}`	Dict comprehension
`{x for x in lst}`	Set comprehension
`(x for x in lst)`	Generator expression (lazy)

Useful Built-ins

Frequently used Python built-in functions.

`print(x)`	Print to stdout
`input("prompt")`	Read user input as string
`len(x)`	Length of sequence or collection
`range(n) / range(a, b, step)`	Integer sequence
`enumerate(lst)`	(index, value) pairs
`zip(a, b)`	Pair elements from two iterables
`map(func, lst)`	Apply function to each element
`filter(func, lst)`	Filter elements by predicate
`sorted(lst)`	Return sorted copy
`sum(lst), min(lst), max(lst)`	Sum, minimum, maximum
`abs(x), round(x, n)`	Absolute value, round to n places
`int(x), float(x), str(x)`	Type conversion

Related Guides

Python f-Strings	String formatting with f-strings
Python Lists	List operations and methods
Python Dictionaries	Dictionary operations and methods
Python for Loop	Iterating with for loops
Python while Loop	Looping with while
Python if/else Statement	Conditionals and branching
How to Replace a String in Python	String replacement methods
How to Split a String in Python	Splitting strings into lists