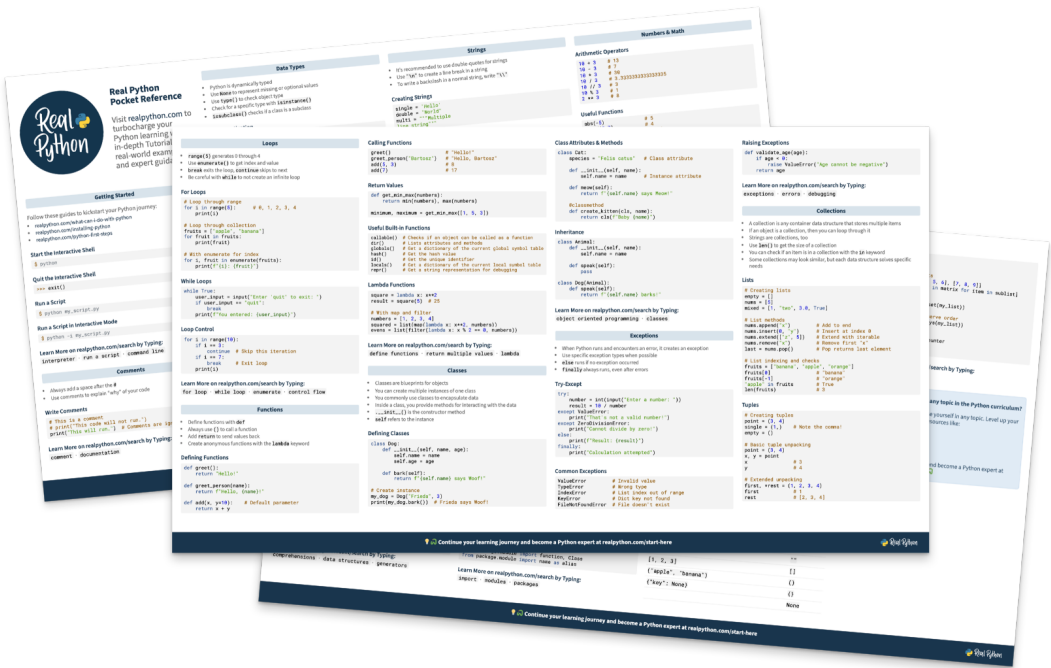


Python Cheat Sheet

This page contains a condensed overview of the Python programming language. It covers Python setup, syntax, data types, variables, strings, control flow, functions, classes, errors, I/O, and more! You can also download the information as a printable cheat sheet:

Free Bonus: Python Cheat Sheet

Get a **Python Cheat Sheet (PDF)** and learn the basics of Python 3, like working with data types, dictionaries, lists, and Python functions:



Email Address

Send My Python Cheat Sheet »

Continue exploring realpython.com to turbocharge your Python learning with in-depth tutorials, real-world examples, and expert guidance.

Getting Started

Shell

Start the Interactive Shell

```
$ python
```

Python

Quit the Interactive Shell

```
>>> exit()
```

Shell

Run a Script

```
$ python my_script.py
```

Shell

Run a Script in Interactive Mode

```
$ python -i my_script.py
```

Resources

Show/Hide

Comments

- Always add a space after the #
- Use comments to explain “why” of your code

Python

Write Comments

```
# This is a comment
# print("This code will not run.")
print("This will run.") # Comments are ignored by Python
```

Resources

Show/Hide

Data Types

- Python is dynamically typed
- Use None to represent missing or optional values
- Use type() to check object type
- Check for a specific type with isinstance()
- subclass() checks if a class is a subclass

Python

Type Investigation

```
type(42)           # <class 'int'>
type(3.14)         # <class 'float'>
type("Hello")     # <class 'str'>
type(True)        # <class 'bool'>
type(None)        # <class 'NoneType'>

isinstance(3.14, float) # True
issubclass(int, object) # True - everything inherits from object
```

Python

Type Conversion

```
int("42")          # 42
float("3.14")      # 3.14
str(42)            # "42"
bool(1)            # True
list("abc")        # ["a", "b", "c"]
```

Resources

Show/Hide

Variables & Assignment

- Variables are created when first assigned
- Use descriptive variable names
- Follow snake_case convention

Python

Basic Assignment

```
name = "Leo"       # String
age = 7            # Integer
height = 5.6       # Float
is_cat = True      # Boolean
flaws = None       # None type
```

Python

Parallel & Chained Assignments

```
x, y = 10, 20      # Assign multiple values
a = b = c = 0      # Give same value to multiple variables
```

Python

Augmented Assignments

```
counter += 1
numbers += [4, 5]
permissions |= write
```

Resources

Show/Hide

Strings

- It’s recommended to use double-quotes for strings
- Use "\n" to create a line break in a string
- To write a backslash in a normal string, write "\\"

Python

Creating Strings

```
single = 'Hello'
double = "World"
multi = """Multiple
line string"""
```

Python

String Operations

```
greeting = "me" + "ow!" # "meow!"
repeat = "Meow!" * 3     # "Meow!Meow!Meow!"
length = len("Python")  # 6
```

Python

String Methods

```
"a".upper()      # "A"
"A".lower()      # "a"
" a ".strip()    # "a"
"abc".replace("bc", "ha") # "aha"
"a b".split()    # ["a", "b"]
"-".join(["a", "b"]) # "a-b"
```

Python

String Indexing & Slicing

```
text = "Python"
text[0]      # "P" (first)
text[-1]     # "n" (last)
text[1:4]    # "yth" (slice)
text[:3]     # "Pyt" (from start)
text[3:]     # "hon" (to end)
text[::2]    # "Pto" (every 2nd)
text[::-1]   # "nohtyP" (reverse)
```

Python

String Formatting

```
# f-strings
name = "Aubrey"
age = 2
f"Hello, {name}!"      # "Hello, Aubrey!"
f"{name} is {age} years old" # "Aubrey is 2 years old"
f"Debug: {age=}"       # "Debug: age=2"

# Format method
template = "Hello, {name}! You're {age}."
template.format(name="Aubrey", age=2) # "Hello, Aubrey! You're 2."
```

PythonRaw Strings

```
# Normal string with an escaped tab
"This is:\tCool."      # "This is:   Cool."

# Raw string with escape sequences
r"This is:\tCool."     # "This is:\tCool."
```

Resources

Show/Hide

 **Free Bonus:** Python Cheat Sheet (PDF) Click to preview & download

Numbers & Math

PythonArithmetic Operators

```
10 + 3    # 13
10 - 3    # 7
10 * 3    # 30
10 / 3    # 3.3333333333333335
10 // 3   # 3
10 % 3    # 1
2 ** 3    # 8
```

PythonUseful Functions

```
abs(-5)      # 5
round(3.7)   # 4
round(3.14159, 2) # 3.14
min(3, 1, 2) # 1
max(3, 1, 2) # 3
sum([1, 2, 3]) # 6
```

Resources

Show/Hide

Conditionals

- Python uses indentation for code blocks
- Use 4 spaces per indentation level

PythonIf-Elif-Else

```
if age < 13:
    category = "child"
elif age < 20:
    category = "teenager"
else:
    category = "adult"
```

PythonComparison Operators

```
x == y    # Equal to
x != y    # Not equal to
x < y     # Less than
x <= y    # Less than or equal
x > y     # Greater than
x >= y    # Greater than or equal
```

PythonLogical Operators

```
if age >= 18 and has_car:
    print("Roadtrip!")

if is_weekend or is_holiday:
    print("No work today.")

if not is_raining:
    print("You can go outside.")
```

Resources

Show/Hide

Loops

- range(5) generates 0 through 4
- Use enumerate() to get index and value
- break exits the loop, continue skips to next
- Be careful with while to not create an infinite loop

Python

For Loops

```
# Loop through range
for i in range(5):      # 0, 1, 2, 3, 4
    print(i)

# Loop through collection
fruits = ["apple", "banana"]
for fruit in fruits:
    print(fruit)

# With enumerate for index
for i, fruit in enumerate(fruits):
    print(f"{i}: {fruit}")
```

Python

While Loops

```
while True:
    user_input = input("Enter 'quit' to exit: ")
    if user_input == "quit":
        break
    print(f"You entered: {user_input}")
```

Python

Loop Control

```
for i in range(10):
    if i == 3:
        continue # Skip this iteration
    if i == 7:
        break     # Exit loop
    print(i)
```

Resources

Show/Hide

Functions

- Define functions with def
- Always use () to call a function
- Add return to send values back
- Create anonymous functions with the lambda keyword

Python

Defining Functions

```
def greet():  
    return "Hello!"  
  
def greet_person(name):  
    return f"Hello, {name}!"  
  
def add(x, y=10):    # Default parameter  
    return x + y
```

Python

Calling Functions

```
greet()                # "Hello!"  
greet_person("Bartosz") # "Hello, Bartosz"  
add(5, 3)              # 8  
add(7)                 # 17
```

Python

Return Values

```
def get_min_max(numbers):  
    return min(numbers), max(numbers)  
  
minimum, maximum = get_min_max([1, 5, 3])
```

Python

Useful Built-in Functions

```
callable() # Checks if an object can be called as a function  
dir()      # Lists attributes and methods  
globals()  # Get a dictionary of the current global symbol table  
hash()     # Get the hash value  
id()       # Get the unique identifier  
locals()   # Get a dictionary of the current local symbol table  
repr()     # Get a string representation for debugging
```

Python

Lambda Functions

```
square = lambda x: x**2  
result = square(5) # 25  
  
# With map and filter  
numbers = [1, 2, 3, 4]  
squared = list(map(lambda x: x**2, numbers))  
evens = list(filter(lambda x: x % 2 == 0, numbers))
```

Resources

Show/Hide

Classes

- Classes are blueprints for objects
- You can create multiple instances of one class
- You commonly use classes to encapsulate data
- Inside a class, you provide methods for interacting with the data
- `__init__()` is the constructor method
- `self` refers to the instance

Python

Defining Classes

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

    def bark(self):
        return f"{self.name} says Woof!"

# Create instance
my_dog = Dog("Frieda", 3)
print(my_dog.bark()) # Frieda says Woof!
```

Python

Class Attributes & Methods

```
class Cat:
    species = "Felis catus" # Class attribute

    def __init__(self, name):
        self.name = name # Instance attribute

    def meow(self):
        return f"{self.name} says Meow!"

    @classmethod
    def create_kitten(cls, name):
        return cls(f"Baby {name}")
```

Python

Inheritance

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

    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return f"{self.name} barks!"
```

Resources

Show/Hide



Get Your Free Python Cheat Sheet (PDF): Learn the basics, fast.

Email address

Get »

Exceptions

- When Python runs and encounters an error, it creates an exception
- Use specific exception types when possible
- else runs if no exception occurred
- finally always runs, even after errors

Python

Try-Except

```
try:
    number = int(input("Enter a number: "))
    result = 10 / number
except ValueError:
    print("That's not a valid number!")
except ZeroDivisionError:
    print("Cannot divide by zero!")
else:
    print(f"Result: {result}")
finally:
    print("Calculation attempted")
```

Python

Common Exceptions

```
ValueError      # Invalid value
TypeError       # Wrong type
IndexError      # List index out of range
KeyError        # Dict key not found
FileNotFoundError # File doesn't exist
```

Python

Raising Exceptions

```
def validate_age(age):
    if age < 0:
        raise ValueError("Age cannot be negative")
    return age
```

Resources

Show/Hide

Collections

- A collection is any container data structure that stores multiple items
- If an object is a collection, then you can loop through it
- Strings are collections, too
- Use len() to get the size of a collection
- You can check if an item is in a collection with the in keyword
- Some collections may look similar, but each data structure solves specific needs

Python

Lists

```
# Creating lists
empty = []
nums = [5]
mixed = [1, "two", 3.0, True]

# List methods
nums.append("x")      # Add to end
nums.insert(0, "y")   # Insert at index 0
nums.extend(["z", 5]) # Extend with iterable
nums.remove("x")      # Remove first "x"
last = nums.pop()     # Pop returns last element

# List indexing and checks
fruits = ["banana", "apple", "orange"]
fruits[0]             # "banana"
fruits[-1]            # "orange"
"apple" in fruits      # True
len(fruits)           # 3
```

Python

Tuples

```
# Creating tuples
point = (3, 4)
single = (1,)    # Note the comma!
empty = ()

# Basic tuple unpacking
point = (3, 4)
x, y = point
x          # 3
y          # 4

# Extended unpacking
first, *rest = (1, 2, 3, 4)
first        # 1
rest         # [2, 3, 4]
```

Python

Sets

```
# Creating Sets
a = {1, 2, 3}
b = set([3, 4, 4, 5])

# Set Operations
a | b          # {1, 2, 3, 4, 5}
a & b          # {3}
a - b          # {1, 2}
a ^ b          # {1, 2, 4, 5}
```

Python

Dictionaries

```
# Creating Dictionaries
empty = {}
pet = {"name": "Leo", "age": 42}

# Dictionary Operations
pet["sound"] = "Purr!"    # Add key and value
pet["age"] = 7            # Update value
age = pet.get("age", 0)   # Get with default
del pet["sound"]          # Delete key
pet.pop("age")            # Remove and return

# Dictionary Methods
pet = {"name": "Frieda", "sound": "Bark!"}
pet.keys()               # dict_keys(['name', 'sound'])
pet.values()             # dict_values(['Frieda', 'Bark!'])
pet.items()              # dict_items([('name', 'Frieda'), ('sound', 'Bark!')])
```

Resources

Show/Hide

Comprehensions

- You can think of comprehensions as condensed for loops
- Comprehensions are faster than equivalent loops

Python

List Comprehensions

```
# Basic
squares = [x**2 for x in range(10)]

# With condition
evens = [x for x in range(20) if x % 2 == 0]

# Nested
matrix = [[i*j for j in range(3)] for i in range(3)]
```

Python

Other Comprehensions

```
# Dictionary comprehension
word_lengths = {word: len(word) for word in ["hello", "world"]}

# Set comprehension
unique_lengths = {len(word) for word in ["who", "what", "why"]}

# Generator expression
sum_squares = sum(x**2 for x in range(1000))
```

Resources

Show/Hide



Get Your Free Python Cheat Sheet (PDF): Learn the basics, fast.

Get »

File I/O

Python

File Operations

```
# Read an entire file
with open("file.txt", mode="r", encoding="utf-8") as file:
    content = file.read()

# Read a file line by line
with open("file.txt", mode="r", encoding="utf-8") as file:
    for line in file:
        print(line.strip())

# Write a file
with open("output.txt", mode="w", encoding="utf-8") as file:
    file.write("Hello, World!\n")

# Append to a File
with open("log.txt", mode="a", encoding="utf-8") as file:
    file.write("New log entry\n")
```

Resources

Show/Hide

Imports & Modules

- Prefer explicit imports over `import *`
- Use aliases for long module names
- Group imports: standard library, third-party libraries, user-defined modules

Python

Import Styles

```
# Import entire module
import math
result = math.sqrt(16)

# Import specific function
from math import sqrt
result = sqrt(16)

# Import with alias
import numpy as np
array = np.array([1, 2, 3])

# Import all (not recommended)
from math import *
```

Python

Package Imports

```
# Import from package
import package.module
from package import module
from package.subpackage import module

# Import specific items
from package.module import function, Class
from package.module import name as alias
```

Resources

Show/Hide

Virtual Environments

- Virtual Environments are often called “venv”
- Use venvs to isolate project packages from the system-wide Python packages

Shell

Create Virtual Environment



```
$ python -m venv .venv
```

Windows PowerShell

Activate Virtual Environment (Windows)



```
PS> .venv\Scripts\activate
```

Shell

Activate Virtual Environment (Linux & macOS)



```
$ source .venv/bin/activate
```

Shell

Deactivate Virtual Environment



```
(.venv) $ deactivate
```

Resources

Show/Hide

Packages

- The official third-party package repository is the [Python Package Index \(PyPI\)](#).

Shell

Install Packages



```
$ python -m pip install requests
```

Shell

Save Requirements & Install from File



```
$ python -m pip freeze > requirements.txt
$ python -m pip install -r requirements.txt
```

Resources

Show/Hide

Miscellaneous

Truthy	Falsy
-42	0
3.14	0.0
"John"	""
[1, 2, 3]	[]
("apple", "banana")	()
{"key": None}	{}
	None

Python

Pythonic Constructs

```
# Swap variables
a, b = b, a

# Flatten a list of lists
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flat = [item for sublist in matrix for item in sublist]

# Remove duplicates
unique_unordered = list(set(my_list))

# Remove duplicates, preserve order
unique = list(dict.fromkeys(my_list))

# Count occurrences
from collections import Counter

counts = Counter(my_list)
```

Resources

Show/Hide

Do you want to go deeper on any topic in the Python curriculum?

At Real Python you can immerse yourself in any topic. Level up your skills effectively with curated resources like:

- [Learning paths](#)
- [Video courses](#)
- [Written tutorials](#)
- [Interactive quizzes](#)
- [Podcast interviews](#)
- [Reference articles](#)

Continue your learning journey and become a Python expert at realpython.com/start-here 💡🐍

You can download the information as a printable cheat sheet:

Free Bonus: Python Cheat Sheet

Get a **Python Cheat Sheet (PDF)** and learn the basics of Python, like working with data types, dictionaries, lists, and Python functions:



Email Address

Send My Python Cheat Sheet »



High Quality Python Video Courses

Watch Now »

Learn Python

- [Start Here](#)
- [Learning Resources](#)
- [Code Mentor](#)
- [Python Reference](#)
- [Python Cheat Sheet](#)
- [Support Center](#)

Courses & Paths

- [Learning Paths](#)
- [Quizzes & Exercises](#)
- [Browse Topics](#)
- [Workshops](#)
- [Books](#)

Community

- [Podcast](#)
- [Newsletter](#)
- [Community Chat](#)
- [Office Hours](#)
- [Learner Stories](#)

Membership

- [Plans & Pricing](#)
- [Team Plans](#)
- [For Business](#)
- [For Schools](#)
- [Reviews](#)

Company

- [About Us](#)
- [Team](#)
- [Mission & Values](#)
- [Sponsorships](#)
- [Careers](#)
- [Press Kit](#)
- [Merch](#)



[Privacy Policy](#) · [Terms of Use](#) · [Security](#) · [Contact](#)

♥ Happy Pythoning!

© 2012–2025 DevCademy Media Inc. DBA Real Python. All rights reserved.
REALPYTHON™ is a trademark of DevCademy Media Inc.

