

Python 3 Quick Reference

Basics through pandas, requests, csv, json

Basics

Variables

```
name = "Alice" # str
age = 20 # int
gpa = 3.85 # float
active = True # bool
```

Data Types

```
str Text: "hello"
int Integer: 42
float Decimal: 3.14
bool True / False
list Ordered, mutable: [1, 2, 3]
tuple Ordered, immutable: (1, 2)
dict Key-value: {"a": 1}
set Unique items: {1, 2, 3}
```

Arithmetic

```
+ - * Add, subtract, multiply
/ Division (float): 7/2 → 3.5
// Floor division: 7//2 → 3
% Modulo: 7%2 → 1
** Power: 2**3 → 8
```

Type Conversion

```
int("42") # 42
float("3.14") # 3.14
str(100) # "100"
list("abc") # ['a', 'b', 'c']
```

User Input

```
name = input("Your name? ")
age = int(input("Age? "))
```

Strings

Creating Strings

```
s1 = 'single quotes'
s2 = "double quotes"
s3 = """triple quotes
for multiline"""
```

f-Strings (Python 3.6+)

```
name = "Alice"
f"Hello, {name}!" # Hello, Alice!
f"{2 + 3}" # 5
f"{3.14159:.2f}" # 3.14
f"{1000:,"} # 1,000
```

String Slicing

```
s = "Python"
# Index: 0 1 2 3 4 5
s[0] # 'P'
s[-1] # 'n'
s[2:5] # 'tho'
s[:2] # 'Py'
s[2:] # 'thon'
s[::-1] # 'nohtyP' (reverse)
```

String Methods

```
len(s) Length of string
s.upper() UPPERCASE
s.lower() lowercase
s.strip() Remove leading/trailing whitespace
s.split(",") Split into list
",".join(lst) Join list into string
s.replace(a, b) Replace a with b
s.find("x") Index of first match (-1 if none)
s.startswith(x) Check prefix → bool
s.endswith(x) Check suffix → bool
s.count(x) Count occurrences
"x" in s Contains check → bool
```

Lists

Creating & Accessing

```
fruits = ["apple", "banana", "cherry"]
fruits[0] # "apple"
fruits[-1] # "cherry"
fruits[1:3] # ["banana", "cherry"]
```

List Comprehension

```
squares = [x**2 for x in range(5)]
# [0, 1, 4, 9, 16]
evens = [x for x in range(10) if x%2==0]
# [0, 2, 4, 6, 8]
```

List Methods

```
lst.append(x) Add to end
lst.extend(lst2) Add all from lst2
lst.insert(i, x) Insert at index i
lst.pop() Remove & return last
lst.pop(i) Remove & return at i
lst.remove(x) Remove first x
del lst[i] Delete by index
lst.sort() Sort in place
sorted(lst) Return sorted copy
lst.reverse() Reverse in place
len(lst) Number of items
x in lst Membership check
lst.index(x) First index of x
lst.count(x) Count of x
```

Tuples & Sets

Tuples (Immutable)

```
point = (3, 4)
x, y = point # unpacking
point[0] # 3 (read-only)
```

Sets (Unique Items)

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

Dictionaries

Creating & Accessing

```
student = {"name": "Alice", "age": 20}
student["name"] # "Alice"
student.get("gpa", 0) # 0 (default)
student["gpa"] = 3.85 # add/update
```

Dict Comprehension

```
sq = {x: x**2 for x in range(5)}
# {0:0, 1:1, 2:4, 3:9, 4:16}
```

Iterating

```
for k, v in student.items():
    print(f"{k}: {v}")
```

Dict Methods

```
d.keys() All keys
d.values() All values
d.items() All (key, value) pairs
d.get(k, default) Get with fallback
d.update(d2) Merge d2 into d
d.pop(k) Remove & return value
del d[k] Delete key
"k" in d Key exists? → bool
len(d) Number of entries
```

Control Flow

if / elif / else

```
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
else:
    grade = "C"
```

Ternary Operator

```
status = "pass" if score >= 60 else "fail"
```

Loops

for Loop

```
for fruit in ["apple", "banana"]:
    print(fruit)
```

range()

```
range(5) # 0, 1, 2, 3, 4
range(2, 5) # 2, 3, 4
range(0, 10, 2) # 0, 2, 4, 6, 8
```

while Loop

```
count = 0
while count < 10:
    count += 1
```

enumerate() & zip()

```
for i, val in enumerate(["a", "b"]):
    print(i, val) # 0 a, 1 b
```

```
for a, b in zip([1, 2], ["x", "y"]):
    print(a, b) # 1 x, 2 y
```

break & continue

```
for x in range(10):
    if x == 5: break # stop loop
    if x % 2 == 0: continue # skip
```

Python 3 Quick Reference

Functions

Defining & Calling

```
def greet(name, greeting="Hi"):  
    return f"{greeting}, {name}!"  
  
greet("Alice") # "Hi, Alice!"  
greet("Bob", "Hello") # "Hello, Bob!"
```

Multiple Return Values

```
def min_max(lst):  
    return min(lst), max(lst)  
lo, hi = min_max([3, 1, 4, 1, 5])
```

*args & **kwargs

```
def total(*args): # args is a tuple  
    return sum(args)  
total(1, 2, 3) # 6  
  
def info(**kwargs): # kwargs is a dict  
    print(kwargs)
```

Lambda Functions

```
square = lambda x: x**2  
square(5) # 25  
sorted(lst, key=lambda x: x["age"])
```

Classes

```
class Dog:  
    def __init__(self, name, breed):  
        self.name = name  
        self.breed = breed  
  
    def bark(self):  
        return f"{self.name} says Woof!"  
  
dog = Dog("Rex", "Lab")  
dog.bark() # "Rex says Woof!"
```

Inheritance

```
class Puppy(Dog):  
    def __init__(self, name, breed, toy):  
        super().__init__(name, breed)  
        self.toy = toy
```

Error Handling

```
try:  
    result = 10 / 0  
except ZeroDivisionError as e:  
    print(f"Error: {e}")  
finally:  
    print("Always runs")
```

File I/O

Reading Files

```
with open("data.txt") as f:  
    content = f.read() # full text  
  
with open("data.txt") as f:  
    for line in f: # line by line  
        print(line.strip())
```

Writing Files

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

"r" = read "w" = write (overwrite) "a" = append

refmint.com

CSV

```
import csv  
  
with open("data.csv") as f:  
    reader = csv.DictReader(f)  
    for row in reader:  
        print(row["name"])  
  
with open("out.csv", "w", newline="") as f:  
    writer = csv.writer(f)  
    writer.writerow(["name", "age"])
```

JSON

```
import json  
  
data = json.loads('{"name": "Alice"}') # parse  
text = json.dumps(data) # serialize  
  
with open("data.json") as f:  
    data = json.load(f) # read file  
with open("out.json", "w") as f:  
    json.dump(data, f, indent=2) # write file
```

HTTP Requests

```
import requests  
  
# GET  
r = requests.get("https://api.example.com/data")  
r.status_code # 200  
data = r.json() # parse JSON  
  
# POST  
r = requests.post(url, json={"key": "val"})
```

pandas Basics

```
import pandas as pd  
df = pd.read_csv("data.csv")  
df.head() # first 5 rows  
df.shape # (rows, cols)  
df["name"] # single column  
df[df["age"] > 20] # filter rows
```

Useful Built-ins

print()	Output to console
len()	Length / count
type()	Type of object
range()	Sequence of numbers
enumerate()	Index + value pairs
zip()	Pair items from iterables
sorted()	Return sorted copy
sum() min() max()	Aggregate functions

Modules

```
import math  
from math import sqrt, pi  
import pandas as pd # alias
```