Python Notes for Beginners

Basics to OOP, with tested examples, cheat sheet and practice questions

Engineers R World | engineersrworld.com/python

Free for students. Share freely with credit. All code tested on Python 3.13.

Contents

1. Introduction and Setup
2. Variables and Data Types
3. Operators
4. Conditions and Loops
5. Strings
6. Lists, Tuples, Sets and Dictionaries
7. Functions and Modules
8. File Handling
9. Exception Handling
10. Object-Oriented Programming
11. NumPy and Pandas Basics
12. Cheat Sheet
13. Practice Questions

1. Introduction and Setup

Python is a high-level, interpreted, general-purpose programming language created by Guido van Rossum and first released in 1991. It is known for readable syntax and is used in data science, AI, web development, automation and engineering.

- Install Python from python.org and tick "Add Python to PATH" during installation.
- Use VS Code, PyCharm, IDLE or Jupyter Notebook to write code.
- Run a script from the terminal with: python file.py
- Install libraries with pip, for example: pip install numpy pandas matplotlib

```
print("Hello, World!")
```

Output:

```
Hello, World!
```

Comments: a line starting with # is ignored by Python. **Indentation:** Python uses 4 spaces to define code blocks instead of braces.

2. Variables and Data Types

A variable is a name that refers to a value. Python decides the type automatically (dynamic typing). Names may contain letters, digits and underscores, cannot start with a digit, are case-sensitive and cannot be keywords.

Type	Example	Mutable?
int	10, -5	No
float	3.14	No
complex	2 + 3j	No
bool	True, False	No
str	"Python"	No
list	[1, 2, 3]	Yes
tuple	(1, 2, 3)	No
set	{1, 2, 3}	Yes
dict	{"E": 200}	Yes
NoneType	None	No

```
x = 10
y = 3.14
name = "Python"
print(type(x), type(y), type(name))
```

Output:

```
<class 'int'> <class 'float'> <class 'str'>
```

Type conversion: int("25"), float("2.5"), str(100), list("abc"), bool(0).

```
print(int("25") + 5, float("2.5") * 2, str(100) + "%", bool(0))
```

Output:

```
30 5.0 100% False
```

Input: input() always returns a string, so convert it: n = int(input("Enter a number: "))

3. Operators

Type	Operators
Arithmetic	+ - * / // % **
Comparison	== != > < >= <=
Logical	and or not
Assignment	= += -= *= /= //= %= **=
Bitwise	& | ^ ~ << >>
Membership	in not in
Identity	is is not

```
a, b = 17, 5
print(a + b, a - b, a * b, a / b, a // b, a % b, a ** 2)
```

Output:

```
22 12 85 3.4 3 2 289
```

Remember: / always gives a float; // rounds down; ** is right-associative, so 2 ** 3 ** 2 = 512.

4. Conditions and Loops

if, elif, else

```
n = 7
if n % 2 == 0:
    print(n, "is even")
else:
    print(n, "is odd")
```

Output:

```
7 is odd
```

for loop with range()

```
n = 7
for i in range(1, 6):
    print(n, "x", i, "=", n * i)
```

Output:

```
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
```

while loop

```
n = 12345
rev = 0
while n > 0:
    rev = rev * 10 + n % 10
    n //= 10
print(rev)
```

Output:

```
54321
```

break exits a loop, **continue** skips to the next iteration and **pass** is an empty placeholder.

5. Strings

Strings are immutable sequences of characters. Use indexing s[0], negative indexing s[-1] and slicing s[start:stop:step].

```
s = "Engineers"
print(s[::-1])
print("python"[1:4])
```

Output:

```
sreenignE
yth
```

Method	Use
upper(), lower()	Change case
strip()	Remove spaces at both ends
split(), join()	Split into a list / join a list
replace(a, b)	Replace text
find(x)	Position of x (-1 if absent)
startswith(), endswith()	Check start/end
f-strings	f"Stress = {s:.2f} MPa"

6. Lists, Tuples, Sets and Dictionaries

List – ordered, mutable

```
parts = ["bolt", "nut", "washer"]
parts.append("gear")
parts[0] = "screw"
print(parts, len(parts))
```

Output:

```
['screw', 'nut', 'washer', 'gear'] 4
```

List comprehension

```
squares = [x * x for x in range(1, 6)]
print(squares)
```

Output:

```
[1, 4, 9, 16, 25]
```

Tuple – ordered, immutable

```
point = (3, 4)
x, y = point
print(x, y)
```

Output:

```
3 4
```

Set – unique values

```
a = {1, 2, 3, 3}
b = {3, 4}
print(a, a | b, a & b)
```

Output:

```
{1, 2, 3} {1, 2, 3, 4} {3}
```

Dictionary – key: value pairs

```
material = {"name": "Steel", "E_GPa": 200, "density": 7850}
material["yield_MPa"] = 250
print(material["E_GPa"], list(material.keys()))
```

Output:

```
200 ['name', 'E_GPa', 'density', 'yield_MPa']
```

7. Functions and Modules

Define a function with def. Use return to send a value back. Default, keyword, *args and **kwargs arguments make functions flexible.

```
def demo(*args, **kwargs):
    print(args, kwargs)

demo(1, 2, unit="mm")
```

Output:

```
(1, 2) {'unit': 'mm'}
```

Lambda, map and filter

```
nums = [1, 2, 3, 4, 5, 6]
print(list(filter(lambda x: x % 2 == 0, nums)))
print(list(map(lambda x: x * 10, nums)))
```

Output:

```
[2, 4, 6]
[10, 20, 30, 40, 50, 60]
```

Modules

```
import math
print("GCD:", math.gcd(48, 36))
```

Output:

```
GCD: 12
```

A module is a .py file; a package is a folder of modules. Use if __name__ == "__main__": to run code only when the file is executed directly.

8. File Handling

Always open files with with so they close automatically. Modes: "r" read, "w" write (overwrite), "a" append, "rb"/"wb" binary.

```
with open("results.txt", "w") as f:
    f.write("Max stress: 200 MPa\n")

with open("results.txt") as f:
    print(f.read())
```

Output:

```
Max stress: 200 MPa
```

9. Exception Handling

```
def safe_divide(a, b):
    try:
        result = a / b
    except ZeroDivisionError:
        print("Error: division by zero")
        return None
    else:
        print("Division successful")
        return result
    finally:
        print("Done")

print(safe_divide(10, 2))
```

Output:

```
Division successful
Done
5.0
```

Exception	Cause
ZeroDivisionError	Division by zero
ValueError	int("abc")
TypeError	"5" + 5
KeyError	Missing dictionary key
IndexError	Index out of range
FileNotFoundError	File does not exist

10. Object-Oriented Programming

A class is a blueprint; an object is an instance. __init__ is the constructor and self refers to the current object. The four OOP concepts are encapsulation, inheritance, polymorphism and abstraction.

```
class Beam:
    def __init__(self, length, load):
        self.length = length
        self.load = load

    def max_moment(self):
        return self.load * self.length / 4

b = Beam(4, 1000)
print(b.length, b.max_moment())
```

Output:

```
4 1000.0
```

Inheritance

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

    def info(self):
        return f"{self.name} has {self.wheels} wheels"

class Car(Vehicle):
    def __init__(self, name):
        super().__init__(name, 4)

print(Car("Sedan").info())
```

Output:

```
Sedan has 4 wheels
```

11. NumPy and Pandas Basics

```
import numpy as np
force = np.array([1000, 2000, 3000])     # N
area = 50e-6                              # m^2
stress = force / area / 1e6               # MPa
print(stress)
```

Output:

```
[20. 40. 60.]
```

```
import pandas as pd
df = pd.DataFrame({"branch": ["Mech", "Aero", "Mech", "Aero", "Mech"],
                   "marks": [78, 85, 66, 90, 72]})
print(df.groupby("branch")["marks"].mean())
```

Output:

```
branch
Aero    87.5
Mech    72.0
Name: marks, dtype: float64
```

12. Cheat Sheet

Task	Code
Print	print("text", value)
Input as number	n = int(input())
Length	len(x)
Range	range(start, stop, step)
Reverse	x[::-1]
Sort	sorted(x), x.sort()
Max / min / sum	max(x), min(x), sum(x)
Remove duplicates	list(dict.fromkeys(x))
Count items	collections.Counter(x)
Round	round(x, 2)
Format	f"{value:.2f}"
Check type	type(x), isinstance(x, int)
Keywords count	len(keyword.kwlist) # 35

13. Practice Questions

1. Check whether a number is prime.
2. Print the Fibonacci series up to n terms.
3. Reverse a string without slicing.
4. Find the second largest number in a list.
5. Count vowels in a sentence.
6. Check whether two words are anagrams.
7. Merge two dictionaries.
8. Read a text file and count its lines.
9. Create a Student class with a method to return grade from marks.
10. Calculate stress = force / area for five load values using a list comprehension.

Solutions and 34 more programs: **engineersrworld.com/python-programs-with-output**

Test yourself: **engineersrworld.com/python-mcq-questions** | Interview prep: **engineersrworld.com/python-interview-questions**