Python Basics

0

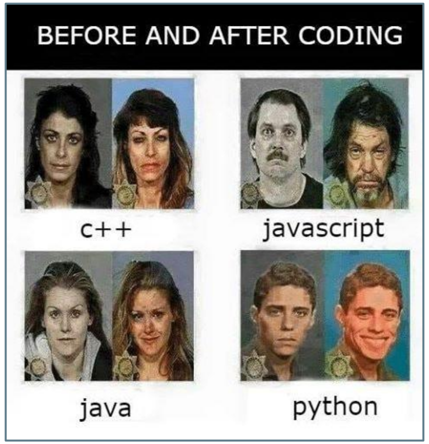
Introduction

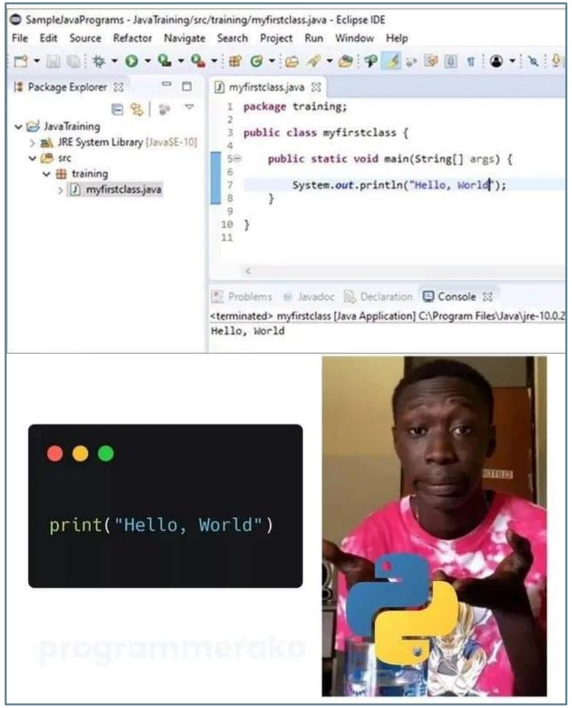
What is Python?

Python is a popular programming language which was created by Guido van Rossum, and released in 1991.

It is used for web development, software development, mathematics, Statistics, system scripting, etc.

Why Python?

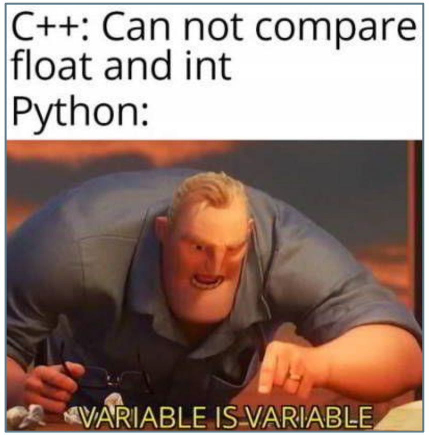



SampleJavaPrograms - JavaTraining/src/training/myfirstclass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
JavaTraining
JRE System Library [JavaSE-10]
src
training
myfirstclass.java
Problems Javadoc Declaration Console
programmerako

```
package training;

public class myfirstclass {

    public static void main(String[] args) {

        System.out.println("Hello, World");
    }

}
```

```
<terminated> myfirstclass [Java Application] C:\Program Files\Java\jre-10.0.2
Hello, World
```

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



Syntax

Variables

```
x = 5
y = 'Hello World!'
```

Comments

```
# This is a comment.
```

Indentation

```
if 5 > 2:
    print('Five is greater than two!')
```

Output and Input

Display output on the screen

```
print('Hello World!')
print('Hello' + 'World!')
print('Hello', 'World!')
print('Hello', 'World', sep = '-', end = '!')
```

Take input from user

```
name = input('Enter your name: ')
print('Hello', name)
```

1

Variables

Variables

Containers for storing data values

```
x = 4
x = 'Sally'
print(x)
```

Assign multiple values

```
x, y, z = 'Orange', 'Banana', 'Cherry'
```

One value to multiple variables

```
x = y = z = 'Orange'
```

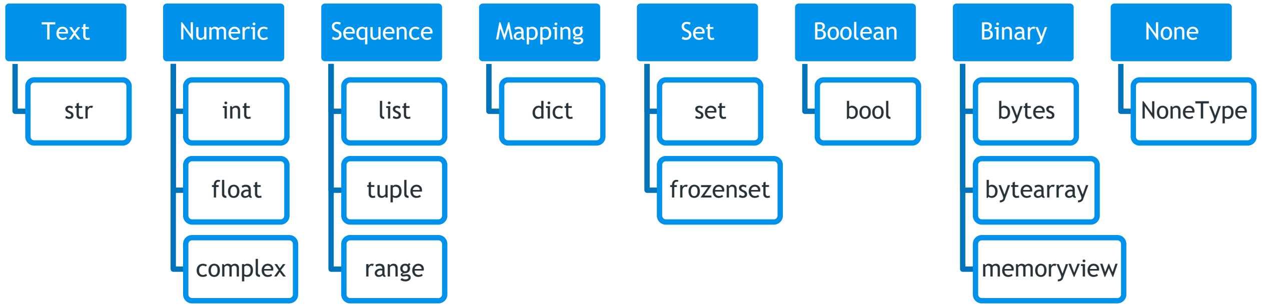
Data Types

Get the type

```
x = -5j
print(type(x))
```

Casting

```
a = str(3)
b = int(3)
c = float(3)
d = bool(3)
```



Numbers

Three numeric types

```
a = 1    # int
b = 2.8  # float
c = 1j   # complex
```

Functions

```
min()  max()  abs()
```

Operators

```
+  -  *  /  %  **  //
```

Strings

Single line and multiline string

```
a = 'Hello!'
b = '''Hello!
How are you?'''
```

F-strings

```
name, age = 'Alireza' , 24
message = f'Hello {name}, you are {age} years old!'
```

Operators

```
+  *
```

Booleans

True or False

```
a = True
b = False
c = 10 >= 9
d = 10 == 9
e = 10 > 9 and 2 + 2 == 5
e = 10 > 9 or 2 + 2 == 5
f = not 2 + 2 == 5
```

Operators

```
<  <=  >  >=  ==  !=  and  or  not
```

Collections

Lists

```
fruits = ['apple', 'banana', 'cherry']
```

Tuples

```
fruits = ('apple', 'banana', 'cherry')
```

Sets

```
fruits = {'apple', 'banana', 'cherry’}
```

Dictionaries

```
fruits = {'apple': 'green', 'banana': 'yellow', 'cherry': 'red'}
```

Lists

Ordered, changeable, and allow duplicate values

```
fruits = ['apple', 'banana', 'cherry', 'apple', 'cherry']
print(fruits[0], fruits[-3], fruits[1:3], fruits[:-1], fruits[0:-1:2], fruits[::-1])
print('orange' in fruits)
print(len(fruits))
```

Update Lists

```
fruits[3] = 'orange'
fruits.append('kiwi')
fruits.remove('banana')
fruits.pop(3)
```

Tuples

Ordered, unchangeable, and allow duplicate values

```
fruits = ('apple', 'banana', 'cherry', 'apple', 'cherry')
print(fruits[0], fruits[-3], fruits[1:3], fruits[:-1], fruits[0:-1:2], fruits[::-1])
print('orange' in fruits)
print(len(fruits))
```

Update tuples

```
fruitsList = list(fruits)
fruitsList[3] = 'orange'
fruitsList.remove('banana')
fruits = tuple(fruitsList)
```

Sets

Unordered, unchangeable, and no duplicate values

```
fruits = {'apple', 'banana', 'cherry'}
print('orange' in fruits)
print(len(fruits))
```

Update sets

```
fruits.add('orange')
fruits.remove('banana')
fruits.pop()
```

Join Sets

Join Sets

```
students = {'Ali', 'Zahra', 'Bahram'}
workers = {'Zahra', 'Bahram', 'Nahid'}
allMembers = students.union(workers)
wortudents = students.intersection(workers)
notWortudents = students.symmetric_difference(workers)
nonworkingStudents = students - workers
nonstudentWorkers = workers - students
```

Dictionaries

Ordered, changeable, and no duplicate values

```
fruits = {'apple': 'green', 'banana': 'yellow', 'cherry': 'red'}
print(fruits['apple'])
print('orange' in fruits)
print(len(fruits))
print(fruits.keys())
print(fruits.values())
```

Update Dictionaries

```
fruits['apple'] = 'red'
fruits['kiwi'] = 'green'
fruits.pop('banana')
```

2

Conditions

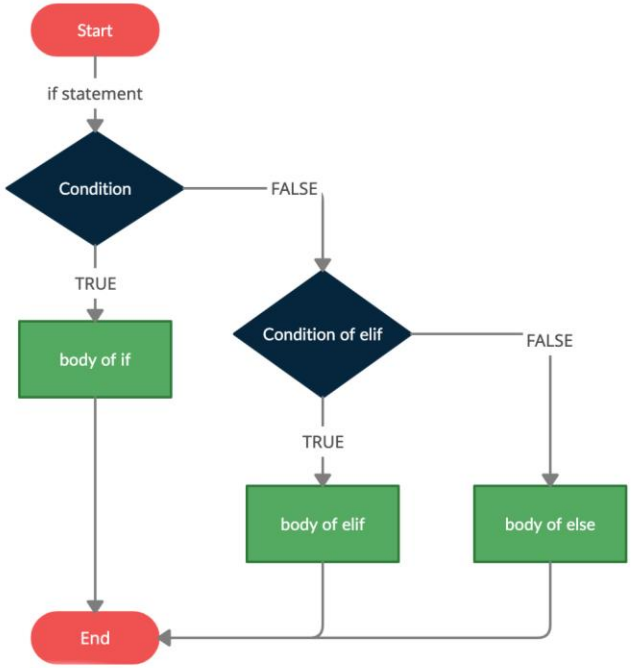
If Statements

If statement

```
a, b = 200, 100
if a > b:
    print('a')
elif a < b:
    print('b')
else:
    print('=')
```

Shorthand if statement

```
a, b = 200, 100
print('a') if a > b else print('b') if a < b else print('=')
```



Nested If Statements

Nested if

```
score = 18
if score >= 10:
    print('Pass')
    if score >= 18:
        print('with grade A!')
    elif score >= 15:
        print('with grade B!')
    elif score >= 12:
        print('with grade C!')
    else:
        print('with grade D!')
else:
    print('Fail!')
```

Match Case

Switch statement

```
language = input('What is the programming language you want to learn? ')
match language:
    case 'JavaScript':
        print('You can become a web developer.')
    case 'Python':
        print('You can become a Data Scientist.')
    case 'PHP':
        print('You can become a backend developer.')
    case 'Solidity':
        print('You can become a Blockchain developer.')
    case _:
        print('The language doesn't matter, what matters is solving problems.')
```

3

Loops

While Loops

Execute a set of statements as long as a condition is true

```
i = 1
while i < 10:
    print(i)
    i += 1
else:
    print(f'Finished after {i} loops')
```

For Loops

Iterating over a collection

```
for fruit in ['apple', 'banana', 'cherry']:
    print(fruit)
else:
    print('Finished!')
```

Iterating over a string

```
for letter in 'apple':
    print(letter)
```

Looping Through a Range

```
for number in range(20):
    print(number)


for number in range(5, 20):
    print(number)


for number in range(5, 20, 2):
    print(number)
```

Break and Continue

Break and continue

```
for i in range(10):
    if i == 2:
        continue
    if i == 5:
        break
    print(i)
else:
    print(f'Finished after {i} loops')
```

Nested Loops

Nested while and for loops

```
number = 2
adjectives = ['big', 'tasty']
fruits = ['apple', 'banana', 'orange']
while number < 5:
    for adjective in adjectives:
        for fruit in fruits:
            print(f'{number} {adjective} {fruit}s')
    number += 1
```

4

Functions

Functions

Blocks of code which only run when they are called

```
def greeter():
    print('Hello!')
greeter()
```

Arguments and return

```
def greeter(firstName, lastName):
    greeting = f'Hello {firstName} {lastName}!'
    return greeting
message = greeter('Alireza', 'Nezhadshamsi')
print(message)
```

Functions

Keyword arguments

```
def greeter(firstName, lastName):
    return f'Hello {firstName} {lastName}!'
print(greeter(lastName = 'Nezhadshamsi', firstName = 'Alireza'))
```

Default parameter value

```
def greeter(firstName = 'dear', lastName = 'user'):
    return f'Hello {firstName} {lastName}!'
print(greeter())
```

Functions

Type Annotations

```
def fullName(firstName: str, lastName: str) -> str:
    return f'{firstName.title()} {lastName.title()}'
print(fullName('alireza', 'nezhadshamsi'))
```

Recursion

```
def factorial(number):
    return number * factorial(number - 1) if number > 1 else 1
print(factorial(4))
```

Lambda

Small anonymous functions

```
greeter = lambda: print('Hello')
print(greeter())
```

Arguments

```
greeter = (lambda name = 'dear user': f'Hello {name}!')
print(greeter(name = 'Alireza'))
```

Lambda

Single Expression

```
(lambda x: (x % 2 and 'odd' or 'even'))(7)
```

Nested lambda

```
funcPlus = lambda x, func: x + func(x)
print(funcPlus(5, lambda x: x * x))
```

5

Errors

Try Except

Test a block of code for errors and handle the error

```
try:
    print(x)
except NameError:
    print('Variable x is not defined')
except:
    print('Something else went wrong')
else:
    print('Nothing went wrong')
```

Exceptions

Handle the error

```
try:
    print(x)
except Exception as e:
    print(e)
```

Raise an error

```
x = -1
if x < 0:
    raise Exception('Please enter a positive number')
```

6

Modules

Modules

A module is a file containing a set of functions you want to include in your application.

To create a module just save the code you want in a file with the file extension .py.

Python module index: https://docs.python.org/3/py-modindex.html

Using a module

```
import myModule as m
m.greeter('Alireza')   # Functions
importedName = m.name  # Variables
```

Import from a module

```
from mymodule import greeter, name
```

Packages

A package contains all the files you need for a module.

PIP is a package manager for Python packages, or modules if you like.

Python package index: https://pypi.org/

Download a package

```
pip install numpy
```

Remove a package

```
pip uninstall numpy
```

List packages

```
pip list
```

Datetime

Import and use datetime

```
import datetime
now = datetime.datetime.now()
print(now)
myBirthday = datetime.datetime(1999, 1, 5)
print(myBirthday.strftime('%x'))
print(myBirthday.strftime('%A'))
print((now - myBirthday).days)
```

Math

Import and use math

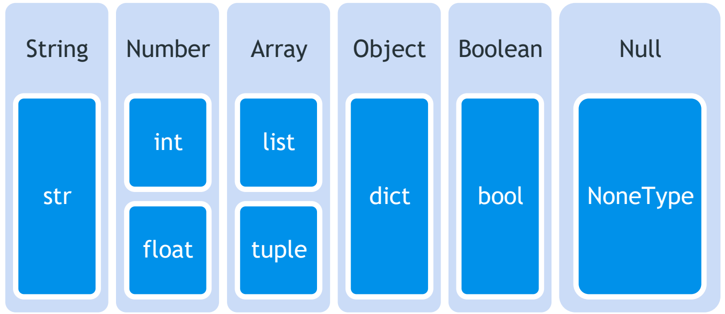
```
import math
print(math.sqrt(64))
print(math.ceil(1.4))
print(math.floor(1.4))
print(math.exp(5))
print(math.factorial(5))
print(math.log(5))
print(math.pi)
print(math.e)
```

JSON

JSON is a syntax for storing and exchanging data.

Import and use math

```
import json
myJson = '{ "name": "Alireza", "age": 24 }'
myDict = json.loads(x)
myDict['age'] = 29
anotherJson = json.dumps(x)
```



String	Number	Array	Object	Boolean	Null
str	int	list	dict	bool	NoneType
	float	tuple			

Requests

Import and use math

```
import requests
response = requests.get('https://fipiran.com', timeout = 1)
print(response)
print(response.status_code)
print(response.encoding)
print(response.text)
```

Thanks!

You can find me at:

github.com/AleeRezaa

t.me/Alee_Rezaa

alee_rezaa@outlook.com