Python Introduction

Python is an object-oriented programming language which is widely used and is very popular. It was created by Guido van Rossum in 1991 and developed by Python Software Foundation. Python can be used to do a variety of tasks like:

* Create Web Applications (Server side)
* File Handling along with database systems
* Machine Learning

Python works on various platforms like Linux, Windows, Mac, Raspberry Pi, etc and has very simple syntax similar to English. These simple syntaxes can actually reduce the lines of code compared to other programming languages.

Another remarkable feature of Python is it runs on interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

The latest version of Python is Python3 and we’ll use this for the entire course.

We can use any text editor to write python code. Also we can use Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Indentation is very important in Python always be careful with that!

Variable Input and Output

The syntax for taking input and output is very simple!

For taking an integer input in a viable:

‘’’

a = int(input(“Enter number”))

‘’’

**NOTE**: Here int() is used to specify the data type ; by default it considers it as a string.

Syntax to output or print the variable:

‘’’

print(a)

‘’’

## Conditions and If statements

Python supports the logical conditions from mathematics:

* Equals: a == b
* Not Equals: a!=b
* Less than: a<b
* Less than or equal to: a<=b
* Greater than: a>b
* Greater than or equal to: a>=b

These conditional statements are generally used with if statements:

Conditional- If

The if statement is written with ‘if’

‘’’

a=10

b=20

if a>b

print(“a is greater than b.”)

‘’’

Conditional- Else-If

The else if statement is written as ‘elif’

‘’’

a = 10  
b = 10  
if b > a:  
 print("b is greater than a")  
elif a == b:  
 print("a and b are equal")

‘’’

Conditional- Else

The else state is written with ‘else’

‘’’

a = 200  
b = 33  
if b > a:  
 print("b is greater than a")  
elif a == b:  
 print("a and b are equal")  
else:  
 print("a is greater than b")

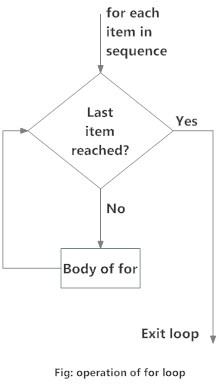
‘’’

Loops in Python

Python programming language provides loops for looping or iterating purposes. Python loops are of 3 types, although they have similar functionality they differ in syntax and condition checking time.

For Loops

It has similar workings compared to other object-oriented programming languages. It iterates over a sequence (ex: List, dictionary, tuple, etc).



‘’’

cars = ["TATA", "Hyundai", "Honda"]

for x in cars:

print(x)

‘’’

OUTPUT:

TATA

Hyundai

Honda

NOTE: The variable x is not required to declare beforehand.

Break & Continue Statements

We can stop a loop with ‘break’ statement without having to go through all the elements in the list.

‘’’

cars = ["TATA", "Hyundai", "Honda"]

for x in cars:

print(x)

if x == " Hyundai ":

break

‘’’

OUTPUT:

TATA

Hyundai

The program breaks out of the loop without iterating through all the elements.

On the other hand, continue statement stops the current iteration and continues with the next iteration.

‘’’

cars = ["TATA", "Hyundai", "Honda"]

for x in cars:

if x == " Hyundai ":

continue

print(x)

‘’’

OUTPUT:

TATA

Honda

range() function in For Loop

To loop through a set of code for specified number of times, we can use the range() function in for loop, The range() function returns a sequence of numbers, starting from 0(default), and increments by 1 (default), and ends at a specified number (n-1).

‘’’

for x in range(5):

print(x)

‘’’

OUTPUT:

0

1

2

3

4

Although by default in range() the loop starts from 0 and increments by 1, we can specify the starting value and increment value ourselves.

‘’’

for x in range(2, 15, 3):

print(x)

‘’’

2

5

8

11

14

Here the starting value is ‘2’ and the incremental value is ‘3’.

While loop in Python

While loop is used to execute a block of code repeatedly until a given condition is met. When the condition becomes false, the line immediately after the loop is executed.

‘’’

sums = 0

while (sums < 4):

sums += 1

print("machine.learning")

‘’’

OUTPUT:

machine.learning

machine.learning

machine.learning

machine.learning

Nested Loops in Python

Just like any other object-oriented programming language, Python allows us to have one loop inside another loop.

‘’’

i = 2

while(i < 50):

j = 2

while(j <= (i/j)):

if not(i%j): break

j = j + 1

if (j > i/j) : print i, " is prime"

i = i + 1

‘’’