Print Statement in Python:

print("Hello, World!")

Values Assigning and printing:

Example:

x = 5  
y = "John"  
print(x)  
print(y)

-----------There are three numeric types in Python:--------------------------------

* int
* float
* complex

x = 1    # int  
y = 2.8  # float  
z = 1j   # complex

print(type(x)) // returns <class 'int'>  
print(type(y)) // returns <class 'float'>  
print(type(z)) // returns <class 'complex'>

---------------------------------------------------PythonCasting--------------------------------------------------------------------

Integer Casting:

x = int(1)   # x will be 1  
y = int(2.8) # y will be 2  
z = int("3") # z will be 3

Float Casting:

x = float(1)     # x will be 1.0  
y = float(2.8)   # y will be 2.8  
z = float("3")   # z will be 3.0  
w = float("4.2") # w will be 4.2

string Casting:

x = str("s1") # x will be 's1'  
y = str(2)    # y will be '2'  
z = str(3.0)  # z will be '3.0'

String Literals:

a = "Hello, World!"  
print(a[1]) //prints e

b = "Hello, World!"  
print(b[2:5]) // prints !

It will print second substring fifth place string

print(a.strip()) //Removes empty places from starting and ending place of a string.

print(len(a)) // length of a

print(a.lower()) // it will convert values to lower case.

print(a.upper()) // it will convert values to Upper case.

print(a.replace("H", "J")) // it will replace H with J .

print(a.split(",")) // It will separate stings based on delimiter.

-------------input the values-------------------------------

x = input() // value will be read and assign to x.

-----------------------------Python Logical Operators:------------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| and | Returns True if both statements are true | x < 5 and  x < 10 | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_logical1) |
| or | Returns True if one of the statements is true | x < 5 or x < 4 | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_logical2) |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_logical3) |

.

* **List** is a collection which is ordered and changeable. Allows duplicate members.

**L=** ["apple", "banana", "cherry"]

* **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.

**T=** ("apple", "banana", "cherry")

Count() : Returns the number of times a specified value occurs in a tuple.

[index(](https://www.w3schools.com/python/ref_tuple_index.asp)) : Searches the tuple for a specified value and returns the position of where it was found

* **Set** is a collection which is unordered and unindexed. No duplicate members.

**S=** {"apple", "banana", "cherry"}

Add() : Adds an element to the set

Clear() : Removes all the elements from the set

[pop()](https://www.w3schools.com/python/ref_set_pop.asp) : Removes an element from the set

remove(): Removes the specified element.

* **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

**D**={  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

--------------------if, if elif else, and,or---------------------------

If:

syntax:

If <condition>:

Statement 1

Statement 2

if elif else :

syntax:

if <condition>:

Statement 1

Statement 2

Elif <condition>:

Statement 1

Statement 2

else:

Statement 1

Statement 2

And:

If <condition1> and <condition2>:

Statement 1

Statement 2

OR:

If <condition1> or <condition2>:

Statement 1

Statement 2

While:

Syntax:

While <condition>

Statement 1

Statement 2

Function:

Sysntax:

def function\_name(parameter):  
  statement 1

To call a function function\_name(parameters)

Function with default parameters:

def my\_function(**country = "Norway"**):

Function with return values:

def my\_function(x):  
  **return 5 \* x**

Array:

To store multiple elements in single place with single data type arrays will help.

Example :

X = cars[23]

Size of the array is 24

Cars[21]=”Reddy”

To add the element to this array.

Cars.append(“Honda”)

Cars.pop()—return and remove the value from this array.

## ---------------------------------Iterator vs Iterable--------------------------------------------

## mytuple = ("apple", "banana", "cherry") myit = iter(mytuple) // iterator applied on this tuple print(next(myit)) // curser moves forward and prints apple print(next(myit)) // curser moves again forward and prints banana print(next(myit)) // // curser moves again again forward and prints cherry

Module: 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

Example

Save this code in a file named mymodule.py

def greeting(name):

print("Hello, " + name)

def greeting(name):

print("Hello, " + name)

Now we can use the module we just created, by using the import statement:

import mymodule

mymodule.greeting("Jonathan")

Built-in Modules:

There are several built-in modules in Python, which you can import whenever you like.

import platform/  
  
x = platform.system()  
print(x)

## ---------------------Parse JSON - Convert from JSON to Python-------------------------------

If you have a JSON string, you can parse it by using the json.loads() method.

import json  
  
# some JSON:  
x =  '{ "name":"John", "age":30, "city":"New York"}'  
  
# parse x:  
y = json.loads(x)  
  
# the result is a Python dictionary:

print(y)

You can convert Python objects of the following types, into JSON strings:

* dictionery
* list
* tuple
* string
* int
* float
* True
* False
* None

Output:

{"name": "John", "age": 30}

["apple", "bananas"]

["apple", "bananas"]

"hello"

42

31.76

true

false

null

import json  
  
print(json.dumps({"name": "John", "age": 30})  
print(json.dumps(["apple", "bananas"]))  
print(json.dumps(("apple", "bananas")))  
print(json.dumps("hello"))  
print(json.dumps(42))  
print(json.dumps(31.76))  
print(json.dumps(True))  
print(json.dumps(False))  
print(json.dumps(None))

## Exception Handling

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.

These exceptions can be handled using the try statement:

Syntax 1:

Try:

Statements //normal statements

except: //catching the exception

Statements

Syntax 2:

Try:

Statements

except:

Statements

Finally: // This block will execute finally

-------------------File Handling ----------------------------------------------

File handling is an important part of any web application.

Python has several functions for creating, reading, updating, and deleting files.

f = open("demofile.txt") // to open a file with default modes.

f = open("demofile.txt", "rt") // oen file with read and text mode

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

f = open("demofile.txt")

print(f.read()) // it will read all text from file

print(f.readline()) // It will read only one line from the text ,that is current line.

f = open("demofile.txt", "a") // open file for append mode  
f.write("Now the file has one more line!") Appending line to end of the file

f = open("demofile.txt", "w") // open a file for writing this will replace with old content.

Delete a File

To delete a file, you must import the OS module, and run its os.remove() function:

import os  
os.remove("demofile.txt")

To remove folder:

import os  
os.rmdir("myfolder")

**Note:** You can only remove empty folders.

os.path.exists("demofile.txt"): // It will check file exist in current path.

---------------------------------------------Python With MySql--------------------------------------------------------------------

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost", #mysql host  
  user="yourusername", #My sql username  
  passwd="yourpassword" #mysql password

**database="mydatabase" # if database not exist its throws an error.**  
)

print(mydb)

mycursor = mydb.cursor()  
  
mycursor.execute("CREATE DATABASE mydatabase") #it will create database.

mycursor.execute("SHOW DATABASES") # This will return list

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))") #create table in current database(mydatabase)

for x in mycursor:   
  print(x)

We use the fetchall() method, which fetches all rows from the last executed statement.

Connect() To established a connection

Cursor() To get cursor object from connection object

Execute() To execute a query on connec

Executescript()

Executemany()

Fetchone()

Fetchmany()

Fetchall()

commit()

roleback()

close()