**Python is a High-level,Interpreted,Scripting language.**

**What is Scripting and why Python is called as a scripting language.**

A scripting language is a programming language designed for interacting and communicating with other programming language.

Ex: Shell,Batch,Perl,Ruby,Python

**Features:**

1. Free and open source
2. Very simple
3. Easy to learn
4. Platform independent
5. Supports both procedural and OOP concepts
6. Supports third party tools

[root@ip-172-31-61-192 ~]# **which python**

/usr/bin/python

**Special characters:**

\n ---new line

\b----back space

\t---tab

\ ----escape

print("a is \"negative\"")

Output---- a is "negative"

A=5

Print(id(a)) -----gives memory id

**Datatypes:**

1. **Numbers:**

Int i=0

Float i=2.5

Complex i=i+2j

>>> a=10

>>> **type(a)**

**<type 'int'>**

**>>>** a=2.5

**>>> type(a)**

**<type 'float'>**

>>> a=2+3j

**>>> type(a)**

**<type 'complex'>**

1. **Strings**: collection of characters

>>> a="Lokesh"

**>>> type(a)**

**<type 'str'>**

>>>

1. **Boolean**

**My-value=True**

**My\_value=False**

1. **Lists:** its like an array where we can store number of heterogeneous elements.

**\***Lists are mutable,meaning we can change the elements

**\***Discrepancy b/w Array and Lists are Array is homogeneous whereas Lists is heterogeneous

>>> a=['A','1',"ABC",'1.5','a']

**>>> type(a)**

**<type 'list'>**

**>>> a[0]**

**'A'**

**>>> a[0:3] ---------------**This is called slicing

**['A', '1', 'ABC']**

**>>> a[0:]**

**['A', '1', 'ABC', '1.5', 'a']**

**>>> a[:4]**

**['A', '1', 'ABC', '1.5']**

**>>>**

1. **Tuples:** Tuples are similar to lists but these are immutable in nature.

>>> a=('1',"ABC",'a','T')

>>> type(a)

**<type 'tuple'>**

>>> a[0]

'1'

**>>> a[1]="BCD"**

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

**TypeError: 'tuple' object does not support item assignment**

1. **Dictionaries:** The elements in dictionaries store in key value pair.

**\*Dictionaries are mutable**

\*Keys have to be unique

>>> a={**1**:'**one**', 2:'two',3:'three'}

>>> type(a)

<type 'dict'>

**>>> a.keys()**

[1, 2, 3]

**>>> a.values()**

['one', 'two', 'three']

>>>

**Input ,Output and eval:**

a=input("Enter first value::")

print(f'value is {a}')

print(type(a))

**Enter first value::10**

**value is 12**

**<class 'str'>**

**Python3:**

a=eval(input("Enter first value::"))  
print(f'value is {a}')  
print(type(a))

**Enter first value:: 12**

**value is 12**

**<class 'int'>**

**Indentation:**

Indentation refers to the spaces at the beginning of a code line,the indentation in Python is very important. Python uses indentation to indicate a block of code.

First program:

**#!/bin/python**

**a=input("enter the value ")**

**if a>0:**

**print " a is positive"**

**else:**

**print " b is negative"**

**OR**

#!/bin/python

print "enter first number"

a=input()

print "enter second number"

b=input()

c=a+b

print c

**Functions:** to perform some task

\*we can call function within the same script

**Module**:: a file which contains definitions and statements.

Collection of similar functions grouped together.

Using which we can call functions in other scripts.

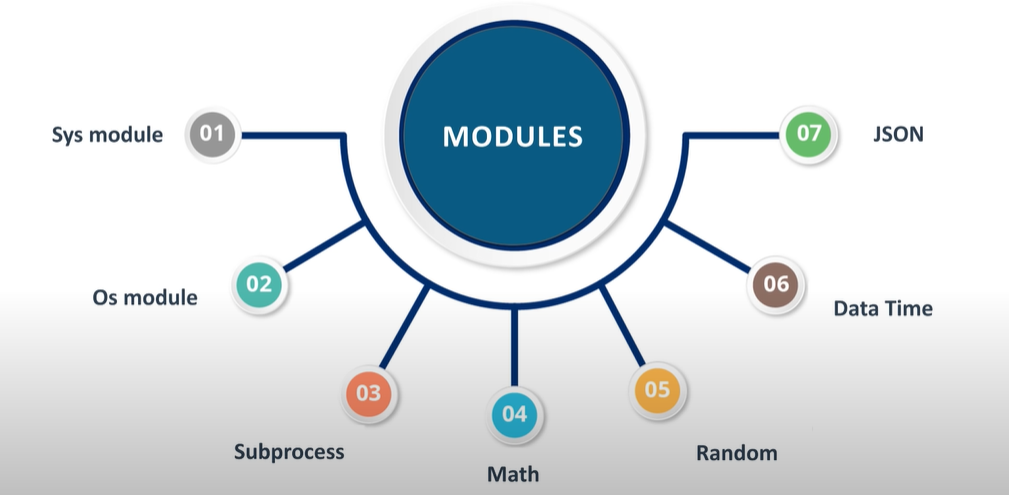
Predefined python script.

Using which we can incorporate some part of other script to the current script for the reusability.

**Import** module\_name

**System modules**---os,sys,str,time,subprocess,pexpect ,ssh,threading,logging

**Modules:**



**Sys module : os related information**

**Import sys**

**Print(sys.version)**

**OS---OS routines(used to interact with OS) ,(creating,deleting ,move folders etc)**

Import os

os.system(“ls –lrt”)

**>>> print(os.getcwd())**

**C:\Users\lokes**

**>>>**

os.chdir(“D:\\lokesh”)

os.mkdir(“D:\\lokesh\\first”)

os.rmdir(“D:\\lokesh\\first”)

os.remove(“D:\\lokesh\\first.py”)

os.path.join(“D:\\lokesh” , “D:\\lokesh\\fisrt”) ----to join two paths

os.path.split(“D:\\lokesh” , “D:\\lokesh\\fisrt”)--- ----to split two paths

os.path.exists(“D:\\lokesh\\first”)

>>> import os

**>>> os.system("pwd")**

/root

0

**>>> os.system("uid")**

sh: uid: command not found

32512

**>>> os.system("free")**

total used free shared buff/cache available

Mem: 1006944 103872 466876 432 436196 761732

Swap: 0 0 0

0

- os.path is one of the modules posixpath, or ntpath

- os.name is 'posix', 'nt', 'os2', 'ce' or 'riscos'

- os.curdir is a string representing the current directory ('.' or ':')

- os.pardir is a string representing the parent directory ('..' or '::')

- os.sep is the (or a most common) pathname separator ('/' or ':' or '\\')

- os.extsep is the extension separator ('.' or '/')

- os.altsep is the alternate pathname separator (None or '/')

- os.pathsep is the component separator used in $PATH etc

- os.linesep is the line separator in text files ('\r' or '\n' or '\r\n')

- os.defpath is the default search path for executables

- os.devnull is the file path of the null device ('/dev/null', etc.)

**-------------------------------------------------------------------------------------------**

**math -------to execute math functions**

**Random------To generate random numbers**

**--------------------------------------------------**

**Date & time:**

**JSON ----to generate json file**

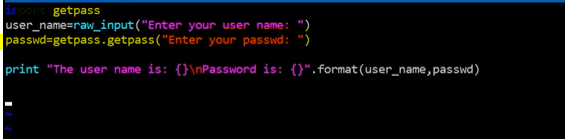
**Platform**

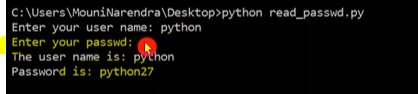
**Subprocess:** enables to interact with OS ,to create processes and alsoTo execute the commands on remote machine

**functions:**

- all functions from posix, nt, os2, or ce, e.g. unlink, stat, etc.

**1)getpass –**to hide password while reading



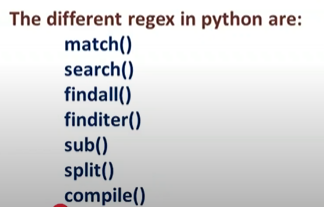


**2) regex or regular expression ---used to look for a special pattern(splitting)**





**Functions in regex**



**time -** This module provides various functions to manipulate time values.import time

Functions:

time() -- return current time in seconds since the Epoch as a float

clock() -- return CPU time since process start as a float

sleep() -- delay for a number of seconds given as a float

gmtime() -- convert seconds since Epoch to UTC tuple

localtime() -- convert seconds since Epoch to local time tuple

asctime() -- convert time tuple to string

ctime() -- convert time in seconds to string

mktime() -- convert local time tuple to seconds since Epoch

strftime() -- convert time tuple to string according to format specification

strptime() -- parse string to time tuple according to format specification

tzset() -- change the local timezone

---------------------------------------------------------------------

**commands** - Execute shell commands via os.popen() and return status, output.

FUNCTIONS

**getoutput(cmd)**

Return output (stdout or stderr) of executing cmd in a shell.

**getstatus(file)**

Return output of "ls -ld <file>" in a string.

**getstatusoutput(cmd)**

Return (status, output) of executing cmd in a shell.

------------------------------------------------------------------