Sys Module:-

> Information about constant ,functions and the methods of interpreter

> Responsible for controlling and interacting with interpretor

> any information on operating system

> example: Piecharm on idle

> some files as sysmodule

> import sys

print(sys.version)

> ex: command line argument

print(sys.argv[])

> argv is list which contains the command line argument passed to a script

------

Various functions:-

stdress: Store error messages

stdin: Accept input from the user

stdout: Print to the screen

end: Quit or end the script

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

2)

os module:

> interacting with os

> primary use of this is

-- create folders

-- remove folders

-- move folders

> Sometimes it is used for change in the working directory

> also access the name of your file path <list dir>

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

Operation with OS module

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

osmodule.py

import os

print(os.getcwd())

=> for current working directory

print(os.getcwd())

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

Changing directory :

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

DIRECTORY SHOULD BE / OR \\ ONLY WORKS

> os.chdir(<directory name>)

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

import os

os.chdir("<directory name>")

print(os.getcwd())

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

CREATING FOLDER

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

os.mkdir(<directory name>/newfolder)

-----------

import os

os.mkdir(<directory name>/newfolder)

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

REMOVING DIRECTORY

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

os.rmdir("<directory name>")

---------

import os

os.rmdir("<directory name>")

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

REMOVING DIRECTORY WHICH IS NOT EMPTY

-----------

shuitl.rmtree("<directory name>")

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

import os

os.rmtree("<directory name>")

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

REMOVING FILE

os.remove("<file path">)

import shuilt

shutil.rmtree("<file path>")

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

2(a)

OS.PATH

import os

print(os.path.join("Dir","file")

a) Join: It takes one or more parts and joints them by using the current OS path separator

print(os.path.join("Dir","file")

b) Split: for scripting purpouse

print(os.path.split("Dir/file.py")

Import os

print(os.path.split("Dir/file.py")

c) exists: Check whether the path exist or not

mostly in TRUE OR FALSE

print(os.path.exists("<file name>")

import os

print(os.path.exists("<file name>")

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

3)

Sub process:

Let us interact with os to create new processes, past the info into and out of them and then return codes

Used by COIL function

So we can simply write the process call it and it will basically run by a command & and wait for the commands and wait for the commands to finish and then gives back the return code

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

4) Math Module : I dont think we will need that

5) Random Module: random no.. generater

ex:

import random

print(random.rand(50)

ex1:

import random

print(random.rand(0,50,10)

the last one is step size, it generate random number with 10 steps

ex2:

print(random.randint(0,20)

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

6) Date time Module:

import datetime

print(datetime.datetime.today())

ex2:

import datetime

print(datetime.datetime.today())

now=(datetime.datetime.today())

other=(datetime.datetime.(1986.6,15))

print(now-other)

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

Python JSON

JSON (JavaScript Object Notation) is a popular data format used for representing structured data. It's common to transmit and receive data between a server and web application in JSON format.

In Python, JSON exists as a string. For example:

p = '{"name": "Bob", "languages": ["Python", "Java"]}'

It's also common to store a JSON object in a file.

Import json Module

To work with JSON (string, or file containing JSON object), you can use Python's json module. You need to import the module before you can use it.

import json

Parse JSON in Python

The json module makes it easy to parse JSON strings and files containing JSON object.

Example 1: Python JSON string to dict

You can parse a JSON string using json.loads() method. The method returns a dictionary.

import json

person = '{"name": "Bob", "languages": ["English", "Fench"]}'

person\_dict = json.loads(person)

# Output: {'name': 'Bob', 'languages': ['English', 'Fench']}

print( person\_dict)

# Output: ['English', 'French']

print(person\_dict['languages'])

Here, person is a JSON string, and person\_dict is a dictionary.

Example 2: Python read JSON file

You can use json.load() method to read a file containing JSON object.

Suppose, you have a file named person.json which contains a JSON object.

{"name": "Bob",

"languages": ["English", "Fench"]

}

Here's how you can parse this file:

import json

with open('path\_to\_file/person.json') as f:

data = json.load(f)

# Output: {'name': 'Bob', 'languages': ['English', 'Fench']}

print(data)

Here, we have used the open() function to read the json file. Then, the file is parsed using json.load() method which gives us a dictionary named data.

If you do not know how to read and write files in Python, we recommend you to check Python File I/O.

Python Convert to JSON string

You can convert a dictionary to JSON string using json.dumps() method.

Example 3: Convert dict to JSON string

import json

person\_dict = {'name': 'Bob',

'age': 12,

'children': None

}

person\_json = json.dumps(person\_dict)

# Output: {"name": "Bob", "age": 12, "children": null}

print(person\_json)

Here's a table showing Python objects and their equivalent conversion to JSON.

Python JSON Equivalent

dict object

list, tuple array

str string

int, float, int number

True true

False false

None null

Writing JSON to a file

To write JSON to a file in Python, we can use json.dump() method.

Example 4: Writing JSON to a file

import json

person\_dict = {"name": "Bob",

"languages": ["English", "Fench"],

"married": True,

"age": 32

}

with open('person.txt', 'w') as json\_file:

json.dump(person\_dict, json\_file)

In the above program, we have opened a file named person.txt in writing mode using 'w'. If the file doesn't already exist, it will be created. Then, json.dump() transforms person\_dict to a JSON string which will be saved in the person.txt file.

When you run the program, the person.txt file will be created. The file has following text inside it.

{"name": "Bob", "languages": ["English", "Fench"], "married": true, "age": 32}

Python pretty print JSON

To analyze and debug JSON data, we may need to print it in a more readable format. This can be done by passing additional parameters indent and sort\_keys to json.dumps() and json.dump() method.

Example 5: Python pretty print JSON

import json

person\_string = '{"name": "Bob", "languages": "English", "numbers": [2, 1.6, null]}'

# Getting dictionary

person\_dict = json.loads(person\_string)

# Pretty Printing JSON string back

print(json.dumps(person\_dict, indent = 4, sort\_keys=True))

When you run the program, the output will be:

{

"languages": "English",

"name": "Bob",

"numbers": [

2,

1.6,

null

]

}

In the above program, we have used 4 spaces for indentation. And, the keys are sorted in ascending order.

By the way, the default value of indent is None. And, the default value of sort\_keys is False.

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

**Aws Scripting:**

Boto Ec2 Module: provides an interface for Elastic Compute cloud (Ec2) services from AWS

> It Reduces the time required to obtain boot new server instances

pip install boto3

pip install awscli

aws configure

the above steps needs to configure before this

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

import boto3

ec2 = boto3.resource('ec2')

instance = ec2.create\_instances(

ImageId='ami-0f65671a86f061fcd',

MinCount=1,

MaxCount=1,

InstanceType='t2.micro')

print(instance[0].id)

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

Termination:

print(instance[0].id) with this we get the instance id

instance\_id = 'i-0d58e369007908a0d'

instance = ec2.Instance(instance\_id)

response = instance.terminate()

print(response)

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