PYTHON SCRIPTING FOR AUTOMATION

Python is everywhere

SEPTEMBER 7, 2021

TECHCIRCLE

Python virtual environment

On unix/linux

```
python3 -m venv /path/to/new/virtual/environment
```

On Windows

```
c:\>c:\Python35\python -m venv c:\path\to\myenv
```

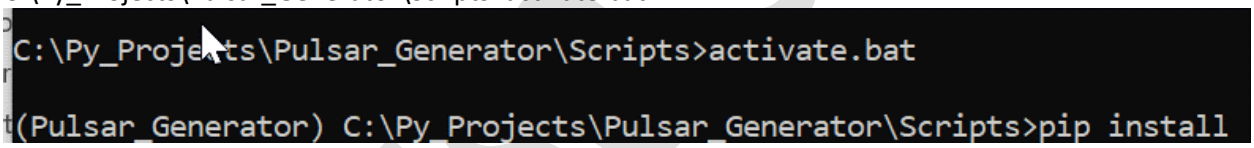
Alternatively, if you configured the `PATH` and `PATHEXT` variables for your Python installation:

```
c:\>python -m venv c:\path\to\myenv
```

To Activate venv

Platform	Shell	Command to activate virtual environment
POSIX	bash/zsh	$ source <venv>/bin/activate
	fish	$ source <venv>/bin/activate.fish
	csh/tcsh	$ source <venv>/bin/activate.csh
	PowerShell Core	$ <venv>/bin/Activate.ps1
Windows	cmd.exe	C:\> <venv>\Scripts\activate.bat
	PowerShell	PS C:\> <venv>\Scripts\Activate.ps1

For an example: cd C:\Py_Projects\Pulsar_Generator\Scripts
And run the batch file inside it
C:\Py_Projects\Pulsar_Generator\Scripts>activate.bat



1-Introduction:

01-Introduction to the Python

02-Python setup on Windows

03-Python setup on Linux (Installing required python on Linux)

04-Editors for Python code

05-How to use Atom Editor to run python script

2-Basics of print, indentation, comments and special characters:

06-simple hello world script

07-Python Indentation

08-Python Comments

09-Usage of special characters with print statement

Type specialchar.py

Print (“Welcome to Python code”)

Print (“Welcome to special char”)

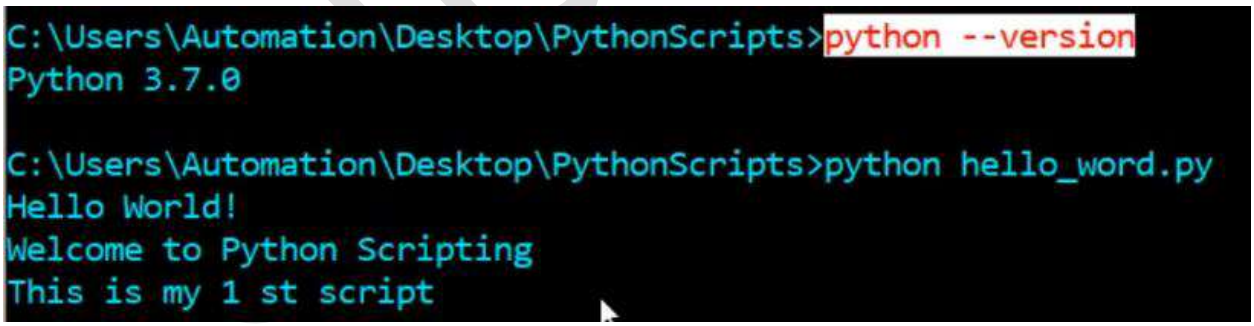
Print (“Welcome to Python code. \n Welcome to special char ”)

Note: Write Special Charecters only inside quotes i.e, ‘ ’, “ “	
Special Char	Usage
/b	your cursor will be back space one position
/b/b	your cursor will be back two positions
/b/b/ b --: print(This is a /b/b/bpython class)	your cursor will be back three positions
/t --: print(This is a /tpython/tclass)	One tab inside your line.
/ --: print(This is a “\python\” class)	Escape
\\ --: print(“C:\\Gouse_1\\Devops\\”)	For windows use \\ double slashes, to call win paths

10-Running Python script on Windows and Linux

On windows:

Set environment variable for windows:



On Unix:

Install python and check the version of python, whether python is properly installed or not.

$apt-get install python

```
$ cat hello_world.py
```

```
#!/usr/local/bin/python3
print("Hello World!")
print("Welcome to Python Scripting")
print("This is my 1 st script")
```

Change permissions to the python script on unix

$chmod +x hello_world.py

```
$ python3 --version

$ python3 hello_world.py
```

```
Hello World!
Welcome to Python Scripting
This is my 1 st script
```

To check the location of your python installed on unix

$ which python

3-Basics of variables and Data Types:

11-Introduction to variables and print with variables

Variables do not need to be declared with any particular *type*, and can even change type after they have been set.

12-Data Types

Variables can store data of different types, and different types can do different things.	
Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

13-working with multiple variable and strings in print

Example	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set

x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

14-Input and Output Syntax

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

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

4-Complete String Operations:

15-Basic operations on strings

Three double quotes:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Three single quotes:

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

Get the character at position 1

```
a = "Hello, World!"
print(a[1])
```

length of a string with len()

```
a = "Hello, World!"
print(len(a))
```

Check if a certain phrase or character is present in a string with keyword in

```
txt = "The best things in life are free!"
print("free" in txt)
```

Check text in string with if statement:

```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

Check text keyword ***not in.***

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

Check text keyword ***not in with if*** *statement:*

```
txt = "The best things in life are free!"
if "expensive" not in txt:
  print("No, 'expensive' is NOT present.")
```

16-case (Lower, Upper etc...) conversion operations

Strings are immutable

Lower case Letters

```
my_string="Python Scripting"
print(my_string.lower())
```

Upper case Letters

```
my_string="Python Scripting"
print(my_string.lower())
print(my_string.upper())
print(my_string)
```

```
python scripting
PYTHON SCRIPTING
Python Scripting
[Finished in 1.6s]
```

Swapcase for Upper to Lower and vise-versa

```
print(my_string.swapcase())
```

```
pYTHON sCRIPTING
```

Title for to show text as a title

```
print(my_string.title())
```

```
Python Scripting Tutorials
[Finished in 0.5s]
```

To get the in-build string functions

txt=”hi”

print(dir(txt))

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

Casefold for convert in lower case.

```
my_string="Python ScriptinG tutorials"
 print(my_string.casefold())
```

```
python scripting tutorials
[Finished in 0.6s]
```

17-Boolean result operations

The result of this string operations is in a Boolean data, with True or False

Startingwith('P') in your condition statements

```
my_str="Python"
print(my_str.startswith('P'))
```

```
print(my_str.startswith('Pyt'))
```

```
True
[Finished in 0.6s]
```

Endingwith('hon')

```
print(my_str.endswith('hon'))
```

```
True
[Finished in 0.6s]
```

islower()

```
my_str="Python"
print(my_str.islower())
```

```
True
[Finished in 0.6s]
```

Isupper()

```
my_str="Python"
print(my_str.isupper())
```

```
False
[Finished in 0.6s]
```

istitle()

```
my_str="Python"
print(my_str.istitle())
```

```
True
[Finished in 0.6s]
```

isspace()

```
my_str="Python Tutorials"
```

It will give --: **False**

```
my_new_str="   "
print(my_new_str.isspace())
print(my_str.isspace())
```

It will give --: **True**

isalpha()

```
my_str="Python Tutorials"
print(my_str.isalpha())
```

It will give --: False

```
my_str="PythonTutorials"
print(my_str.isalpha())
```

It will give --: True

isnumeric()

```
my_str="Python Tutorials"
print(my_str.isnumeric())
```

```
False
[Finished in 0.7s]
```

```
my_str="575854934"
print(my_str.isnumeric())
```

```
True
[Finished in 0.5s]
```

For more help

>>help(str)

18-join, center and zfill (zero fill)

join()

```
>>> x="python"
>>> y="-".join(x)
>>> print(y)
p-y-t-h-o-n
>>> print(x)
python
>>> print("*".join(x))
p*y*t*h*o*n
>>> print("\n".join(x))
p
y
t
h
o
n
>>> print("\t".join(x))
p	y	t	h	o	n
```

Center()

```
>>> my_str="python"
>>> my_new_string="python scripting"
>>> my_str3="string operation"
>>> print(my_str.center(20))
       python
>>> print(f"{my_str.center(20)}\n{my_new_string.cneter(20)}\n{my_str3.center(20)})
```

Here python will take the length first, here we have given 20 and in this 20 it will make our text in middle.

```
       python
  python scripting
  string operation
>>>
```

Zfill() for padding

```
>>> print(my_str.zfill(10))
0000python
```

19-strip, split operations

Strip()

Strip will remove the space by default, if you want to remove other things also you can do.

```
>>> x=" python"
>>> print(x)
 python
>>> x=" python    "
>>> print(x)
 python
>>> print(x.strip())
python
>>> x="python"
>>> print(x.strip('p'))
ython
>>> print(x.strip('n'))
pytho
>>> print(x.strip('+'))
python
>>>
```

Rstrip(), Lstrip()

```
>>> x="python scripting is easy"
>>> print(x.strip('easy'))
python scripting is
>>> x="python scripting is easy python"
>>> print(x.rstrip('python'))
python scripting is easy
>>> print(x.strip('python'))
 scripting is easy
>>> print(x.lstrip('python'))
 scripting is easy python
```

```
>>> x="python./i"
>>> x=x.strip('./i')
>>> print(x)
python
>>> x="pythonyy"
>>> x.strip('p')
'ythonyy'
>>> x.strip('p').strip('y')
'thon'
>>> x.strip('p').rstrip('y')
'ython'
>>> x.strip('p').lstrip('y')
'thonyy'
>>>
```

Split()

```
>>> x="python is easy"
>>> x.split()
['python', 'is', 'easy']
>>> x.split()
['python', 'is', 'easy']
>>> x.split('is')
['python ', ' easy']
>>> x="python is easy and it is very popular"
>>> x.split('is')
['python ', ' easy and it ', ' very popular']
```

20-count, index and find operations on strings

Count()

Is for counting the no.of words and chars in your string

```
>>> x="python is easy and it is popular language"
>>> x.count('is')
2
>>> x.count('p')
3
>>> x.count('t')
2
>>> x.count('a')
5
```

Index()

We have positive+ indexes and negative- indexes

```
>>> x.index('p')
0
>>> x.index('p',1)
25
>>> x.index('p',26)
27
>>> x.index('p',28)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
```

```
>>> x.index('is',23)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: substring not found
>>>
```

find()

Instead of index(), we can use find() you can get +ve and –ve index. If the output is –ve that means the looking search string is not present in after no.

>> First value is a 'search'

>> Second value is search from the index no '2'

```
>>> x.find('p')
0
>>> x.find('p',2)
25
>>> x.find('p',26)
27
>>> x.find('p',28)
-1
```

```
python is easy and it is popular language
>>> java_version="java version 1.6"
>>> java_version.find("java")
0
>>> java_version="Error while finding"
>>> java_version.find("java")
-1
```

21-Practice - Display given string at left_right_center of a line in title format

```
C:\Users\Automation\Desktop\PythonScripts>python allign_text.py
Enter your string: python scripting
python scripting
```

To know in windows coulmns

```
C:\Users\Automation\Desktop\PythonScripts>mode

Status for device CON:
----------------------
    Lines:          9001
    Columns:        122
    Keyboard rate:  31
    Keyboard delay: 1
    Code page:      437
```

```
given_str=input("Enter your string: ")
print(given_str.center(122))
print(given_str.ljust(122))
print(given_str.rjust(122))
```

```
C:\Users\Automation\Desktop\PythonScripts>python allign_text.py
Enter your string: python scriptng
                                                    python scriptng

python scriptng
                                                                                                                    python scriptng
```

In unix to check your terminal columns

$tput cols

For this we need os module to get terminal size

Os module In windows

```
>>> import os
>>> os.get_terminal_size()
os.terminal_size(columns=122, lines=29)
>>> os.get_terminal_size().columns
122
>>>
```

Os module In unix

```
>>> import os
>>> os.get_terminal_size().columns
112
>>> exit
```

Create a script now

```
import os
t_w=os.get_terminal_size().columns

given_str=input("Enter your string: ")
print(given_str.center(t_w))
print(given_str.ljust(t_w))
print(given_str.rjust(t_w))
```

t_w

is a terminal width , python will automatically fetch width size with this t_w

now append with .title

```
import os
t_w=os.get_terminal_size().columns

given_str=input("Enter your string: ")
print(given_str.center(t_w).title())
print(given_str.ljust(t_w).title())
print(given_str.rjust(t_w).title())
```

```
C:\Users\Automation\Desktop\PythonScripts>python allign_text.py
Enter your string: python scripting
                                        Python Scripting

Python Scripting
                                                        Python Scripting
```

5-Data Structures of Python:

22-Introduction to Data Structures and Types of Data Structures

The data structures are the types, where it will store multiple values. Data structure is use to store the collection of data.

```
>>> x=5
>>> print(x)
5
>>> my_values=[3,4,5,'python','devops',5.6]
>>> _
```

There are four built-in data structures.
They are:

- List --> []
- Tuple --> ()
- Dictionary --> {} with key value pair
- Set --> {}

23-Lists (mutable, you can change values)

```
my_list=[]
my_list1=[3,2,4,"python",5.6]

bool(empty_list) ==> False
bool(non-empty_list) ==> True
```

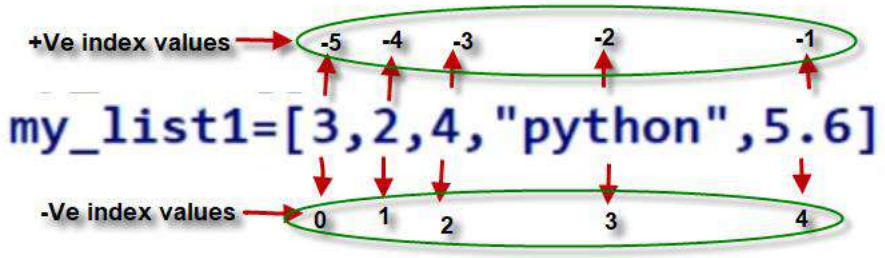
Converting list to Boolean

Boolean of empty list is False
Boolean of non-empty list is True

```
>>> bool(my_list)
False
```

List index values

Calling values in two types, those will be in +ve and –ve formats as follows.




```
print(my_list1[0])
print(my_list1[3])
print(my_list1[-1])
print(my_list1[-2])
```

```
print(my_list1[3][1])
```

```
y
[Finished in 0.9s]
```

Calling list from start and end indexes

```
print(my_list1)
print(my_list1[:])
print(my_list1[0:])
```

```
[3, 2, 4, 'python', 5.6]
[3, 2, 4, 'python', 5.6]
[3, 2, 4, 'python', 5.6]
[Finished in 0.6s]
```

```
print(my_list1[1:4])
```

```
[2, 4, 'python']
[Finished in 0.7s]
```

Find type of variable

```
print(my_list1,type(my_list1))
```

Modify your list values

Because lists are mutable (you can change at any time)

Note

strings and tuples are immutable, you can't change your values.

```
my_list1[0]=45
print(my_list1)
```

```
[45, 2, 4, 'python', 5.6]
[Finished in 0.7s]
```

List doc

```
>>> my_list=[3,4,5]
>>> bool(my_list)
True
>>> my_list
[3, 4, 5]
>>> dir(my_list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__g
e__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '
__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__r
eversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'co
py', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

To get the index of values

```
my_list=[3,5,2,7,3,8,5,9]
print(my_list.index(5))
```

```
6
[Finished in 1.0s]
```

Regex

For getting list values, we have regex concept

Count list value

```
my_list=[3,5,2,5,5,7,3,8,9]
#print(my_list.index(5))
print(my_list.count(5))
```

```
3
[Finished in 0.8s]
```

Clear list values

```
print(my_list)
my_list.clear()
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9]
[]
[Finished in 0.4s]
```

Copy list values from one list to other lists

```
my_list=[3,5,2,7,3,8,9]
```

```
my_new_list=my_list
my_one_list=my_list.copy()
```

Calling address id's of lists

```
print(id(my_list),id(my_new_list))
print(id(my_one_list))
```

```
2175027667592 2175027667592
2175028165576
```

append() is for modifying list

While using append() do print.

```
my_list=[3,5,2,7,3,8,9]
```

```
print(my_list)
my_list.append(56)
```

```
[3, 5, 2, 7, 3, 8, 9, 56]
```

insert() is based on index position

```
my_list.insert(1,45)
print(my_list)
```

```
[3, 45, 5, 2, 7, 3, 8, 9, 56]
[Finished in 0.7s]
```

extend() is result as normal data.

```
my_new_list=[5,6]
my_list.append(my_new_list)
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9, [5, 6]]
[Finished in 0.7s]
```

```
my_new_list=[5,6]
my_list.extend(my_new_list)
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9, 5, 6]
[Finished in 0.7s]
```

Remove() data in a list

Is should be used in if statement.

```
print(my_list)
my_list.remove(10)
  File "C:\Users\Automation
print(my_list)
```

```
print(my_list)
my_list.remove(8)
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9]
[3, 5, 2, 7, 3, 9]
[Finished in 0.7s]
```

Pop() is for removing data from last in the list

```
print(my_list)
#my_list.remove(8)
#print(my_list)
my_list.pop()
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9]
[3, 5, 2, 7, 3, 8]
[Finished in 1.0s]
```

Pop() by index

```
print(my_list)
```

```
my_list.pop(0)
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9]
[5, 2, 7, 3, 8, 9]
[Finished in 0.7s]
```

Reverse()

```
my_list.reverse()
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9]
[9, 8, 3, 7, 2, 5, 3]
[Finished in 0.7s]
```

Sort() ascending

Is for arranging your list data in ascending order (small to big ☺)

```
my_list.sort()
print(my_list)
```

```
[3, 5, 2, 7, 3, 8, 9]
[2, 3, 3, 5, 7, 8, 9]
[Finished in 0.7s]
```

Sort() ascending

To get the descending order with sort()

```
my_list.sort(reverse=True)
```

24-Tuples (immutable , means you can't change the values, once define)

```
#Define a tuple
my_empty=()
my_tuple=(3,4,5)
print(my_empty)
print(my_tuple)
print(bool(my_empty))
print(bool(my_tuple))
```

```
()
(3, 4, 5)
False
True
[Finished in 0.8s]
```

To work with list inside tuple

```
my_tuple=(3,4,[5,6,7],8,9)
print(my_tuple)
print(my_tuple[2][1])
```

```
(3, 4, [5, 6, 7], 8, 9)
6
[Finished in 0.5s]
```

Tuple object can't be changed once you defined.

```
my_tuple[0]=34
```

```
File "C:\Users\Autom
TypeError: 'tuple' object does not support item assignment
```

```
>>> x=(4,5)
>>> dir(x)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getatt
ribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__',
'__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__'
, '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
>>> dir([3,4])
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__g
e__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '
__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__r
eversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'co
py', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

Count()

```
>>> x=(5,6,7,3,0)
>>> x.count(5)
1
>>> x=(5,6,7,3,0,5)
>>> x.count(5)
2
```

Index()

```
>>> x.index(5)
0
>>> x.index(5,1)
5
```

Len()

```
>>> x="hi"
>>> len(x)
2
>>> x=[4,5]
>>> len(x)
2
>>> x=(4,56,9)
>>> len(x)
3
```

Get the values by index

```
my_tuple=(3,4,5,6,78,2,65,0)
print(my_tuple[3:])
```

```
(6, 78, 2, 65, 0)
[Finished in 0.7s]
```

```
print(my_tuple[:6])
```

```
(3, 4, 5, 6, 78, 2)
[Finished in 0.6s]
```

```
print(my_tuple[3:5])
```

```
(6, 78)
[Finished in 0.7s]
```

```
y=5,
x=5,8,9
print(x,type(x))
```

```
(5, 8, 9) <class 'tuple'>
[Finished in 1.0s]
```

25-Dictionaries {'key':'value', 'key':'value'}

```
my_dict={}
print(my_dict,type(my_dict))
```

```
{} <class 'dict'>
[Finished in 0.6s]
```

```
print(bool(my_dict))
```

```
False
[Finished in 0.6s]
```

```
my_dict={'fruit':'apple','animal':'fox',1:'one','two':2}
print(my_dict)
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
[Finished in 0.6s]
```

```
print(my_dict['fruit'])
print(my_dict.get('animal'))
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
apple
fox
[Finished in 1.0s]
```

```
print(my_dict['three'])
```

```
KeyError: 'three'
[Finished in 1.0s with exit code 1]
```

```
print(my_dict.get('three'))
```

```
None
[Finished in 0.9s]
```

```
>>> x={}
>>> dir(x)
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__ge
tattribute__', '__getitem__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__
lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str
__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', '
values']
>>>
```

To add a new value to dictionary{}

```
my_dict['three']=3
print(my_dict)
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2, 'three': 3}
[Finished in 0.6s]
```

To update a new value to existing key value in dictionary{}

```
my_dict['three']=3
print(my_dict)
my_dict['three']=56
print(my_dict)
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2, 'three': 3}
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2, 'three': 56}
[Finished in 0.8s]
```

To get only keys

```
print(my_dict.keys())
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
dict_keys(['fruit', 'animal', 1, 'two'])
[Finished in 0.7s]
```

To get only values

```
print(my_dict.values())
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
dict_keys(['fruit', 'animal', 1, 'two'])
dict_values(['apple', 'fox', 'one', 2])
```

To show your key & values as “item”

```
print(my_dict.items())
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
```

```
dict_items([('fruit', 'apple'), ('animal', 'fox'), (1, 'one'), ('two', 2)])
[Finished in 0.7s]
```

Copy dictionary to new dictionary {}

```
y=my_dict.copy()
print(y)
```

```
y=my_dict.copy()
print(id(y),id(my_dict))
print(y)
```

```
dict_items([('fruit', 'apple'), ('animal', 'fox'), (1, 'one'), ('two', 2)])
1695668001528 1695668001024
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
[Finished in 0.7s]
```

Update()

```
my_new_dict={'four':'4'}
my_dict.update(my_new_dict)
print(my_dict)
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2, 'four': '4'}
[Finished in 0.6s]
```

Pop()

```
my_dict.pop('four')
print(my_dict)
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2, 'four': '4'}
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
[Finished in 0.8s]
```

Popitem() it will remove randomly

```
my_dict.popitem()
print(my_dict)
```

```
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2, 'four': '4'}
{'fruit': 'apple', 'animal': 'fox', 1: 'one', 'two': 2}
{'fruit': 'apple', 'animal': 'fox', 1: 'one'}
[Finished in 0.7s]
```

fromkeys()

```
keys=['a','e','i','o','u']

new_dict=dict.fromkeys(keys)

print(new_dict)
new_dict['a']="first alpha"
print(new_dict)
```

```
{'a': None, 'e': None, 'i': None, 'o': None, 'u': None}
{'a': 'first alpha', 'e': None, 'i': None, 'o': None, 'u': None}
[Finished in 0.5s]
```

Setdefault()

If key is there, then it will don't do anything or else it will create a key and value with setdefault()

```
my_dict={}

my_dict.setdefault('k',45)

print(my_dict)
```

```
{'k': 45}
[Finished in 0.7s]
```

```
my_dict={'fruit':'apple'}
my_dict.setdefault('fruit','orange')
print(my_dict)
```

```
{'k': 45}
{'fruit': 'apple'}
[Finished in 0.7s]
```

From your python 3.7, dictionary is in an order data

```
>>> my_dict={'fruit':'apple','animal':'fox',1:'one','two':2}
>>> print(my_dict)
{1: 'one', 'fruit': 'apple', 'two': 2, 'animal': 'fox'}
```

26-Sets

It will give you a unique data

```
>>> my_set={4,5,7,2,7,0}
>>> print(my_set)
{0, 2, 4, 5, 7}
>>> b  l(my_set)
True
>>> my_set=set({})
>>> print(type(my_set))
<class 'set'>
>>> bool(my_set)
False
>>> my_li=[4,5,6,7]
>>> set(my_li)
{4, 5, 6, 7}
>>> my_li=[4,5,6,7,4,5]
>>> set(my_li)
{4, 5, 6, 7}
>>>
```

```
>>> a={3,4,5,6}
>>> b={5,6,7,8,9}
>>> a.union(b)
{3, 4, 5, 6, 7, 8, 9}
>>> a.intersection(b)
{5, 6}
```

Use the syntax dict[key] = value to append a key - value pair to dict .

```
1. a_dict = {}
2. a_dict["b"] = 4.
3. print(a_dict)
```

6-Operators of Python:

27-Introduction to Operators of Python

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

28-Arithmetic and Assignment Operators

Arithmetic operators are used with numeric values to perform common mathematical operations

Arithmetic Operators		
Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Assignment operators are used to assign values to variables

Assignment Operators		
Operator	**Example**	**Same As**
=	**x = 5**	**x = 5**
+=	**x += 3**	**x = x + 3**
-=	**x -= 3**	**x = x - 3**
***=**	**x *= 3**	**x = x * 3**
/=	**x /= 3**	**x = x / 3**
%=	**x %= 3**	**x = x % 3**
//=	**x //= 3**	**x = x // 3**

**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

29-Comparison Operators

Comparison operators are used to compare two values

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

30-Identity and Membership operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Membership operators are used to test if a sequence is presented in an object

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

31-Logical Operators

Logical operators are used to combine conditional statements

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

32-Bitwise Operators

Bitwise operators are used to compare (binary) numbers

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

7-Conditional statements:

32-Introduction to conditional statements; simple if condition

33-if ... else and if ... elif ... elif ... else condition

34-Practice with conditional statements

Python supports the usual 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 conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the `if` keyword.

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

```
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")
```

8-Working with Python Modules:

35-Introduction to Python Modules

What is a module?

What is a module ?

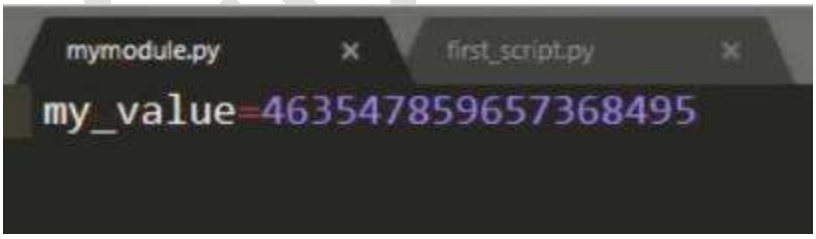
A module is a file containing Python definitions and statements. That means, module containing python functions, classes and variables.

Why module?

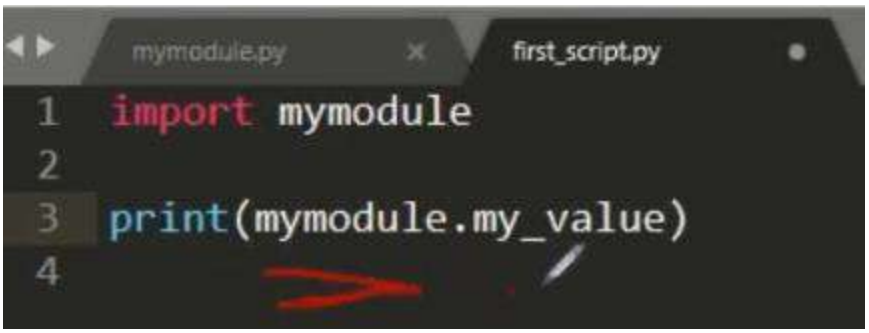
What is the use of module ?

- **Reusability**

Lets create 2 programs



Call mymodule.py in your first_script.py
Note: while calling in your mymodule.py , don't use .py extention.
As on now just place both .py programs in the same location.



To get the list of default modules

```
>>> help("modules")

Please wait a moment while I gather a list of all available modules...

__future__          _warnings           getpass             rlcompleter
_abc                _weakref            gettext             runpy
_ast                _weakrefset         glob                sched
_asyncio            _winapi             gzip                secrets
_bisect             abc                 hashlib             select
_blake2             aifc                heapq               selectors
_bootlocale         antigravity         hello               setuptools
_bz2                any_file            hmac                shelve
_cffi_backend       argparse            html                shlex
_codecs             array               http                shutil
_codecs_cn          asn1crypto          idlelib             signal
_codecs_hk          ast                 imaplib             site
_codecs_iso2022     asynchat            imghdr              six
_codecs_jp          asyncio             imp                 smtpd
_codecs_kr          asyncore            importlib           smtplib
_codecs_tw          atexit              inspect             sndhdr
_collections        audioop             io                  socket
_collections_abc    base64              ipaddress           socketserver
_compat_pickle      bcrypt              itertools           sqlite3
_compression        bdb                 json                sre_compile
_contextvars        binascii            keyword             sre_constants
```

To get the module operations with dir(moduleName)

>>> import <moduleName>

>>> dir(moduleName)

```
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh
', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log1
0', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'tru
nc']
>>>
```

```
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh
', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log1
0', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'tru
nc']
>>> math.pow(2,3)
8.0
>>> math.pi
3.141592653589793
>>>
```

here power() required values, we have given 2 and 3 values

here pi is a variable, so it doesnt required input values

To get documentation with help

>>> help(moduleName)

```
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.

    asinh(x, /)
        Return the inverse hyperbolic sine of x.

    atan(x, /)
        Return the arc tangent (measured in radians) of x.

    atan2(y, x, /)
        Return the arc tangent (measured in radians) of y/x.

-- More  --
```

```
DATA
    e = 2.718281828459045
    inf = inf
    nan = nan
    pi = 3.141592653589793
```

```
>>> math.factorial()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: factorial() takes exactly one argument (0 given)
>>> math.factorial(4)
24
>>> math.factorial(5)
120
```

Pip for installing of 3rd party modules

```
C:\Users\Automation>pip install xlrd
Collecting xlrd
  Using cached https://files.pythonhosted.org/packages/b0/16/63576a1a001752e34bf8ea62e367997530dc553b689356b9879339cf45a4
/xlrd-1.2.0-py2.py3-none-any.whl
Installing collected packages: xlrd
Successfully installed xlrd-1.2.0
You are using pip version 10.0.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\Automation>
```

Suppose, unless you want to work with aws, then use boto3 module

```
C:\Users\Automation>pip install boto3
```

If your want to work on remote servers, you can use paramiko , fabric modules

```
C:\Users\Automation>pip install paramiko
```

Types of modules

Types of Python Modules:

- **Default Modules**
- **Third Party Modules**

Import either default or third party modules before using them.

Differences between “import module” and “from module import *”

```
from math import *
print(pi)
print(pow(3,2))


'''
import math
print(math.pi)
'''
```

```
import math as m
print(m.pi)
```

```
import math
print(math.pi)
print(math.pow(3,2))


Method-2:
=============
import math as m
print(m.pi)
print(m.pow(2,3)

Method-3:
=========
from math import *
print(pi)
print(pow(4,2))

Method-4:
===========
from math import pi,
```

Note: don't prefer Method-4th you will get confuse some times.

How to import multiple modules

```
How to import multiple modules:
import platform
import math
import sys
import os
import subprocess

or
import platform,math,sys,os,subprocess
```

36-platform module

```
import platform

print(platform.system())
```

(The platform module is used to access the underlying platform's data such as hardware, operating system and interpreter version info)

To get the OS name

```
1  import platform
2  print(f"This is {platform.system()} os")
3  print(f'Python version is: {platform.python_version()}')
```

```
system()
    Returns the system/OS name, e.g. 'Linux', 'Windows' or 'Java'.
```

```
import platform
print(f"This is {platform.system()} os")
print(f'Python version is: {platform.python_version()}')
print(platform.python_version_tuple())
```

```
This is Windows os
Python version is: 3.7.0
('3', '7', '0')
[Finished in 0.9s]
```

```
>>> import platform
>>> print(platform.machine())
AMD64
>>> print(platform.release())
10
>>> print(platform.platform())
Windows-10-10.0.17763-SP0
```

```
>>> print(platform.architecture())
('64bit', 'WindowsPE')
>>> print(platform.processor())
Intel64 Family 6 Model 78 Stepping 3, GenuineIntel
>>> print(platform.node())
DESKTOP-CH40V79
>>> print(platform.uname())
uname_result(system='Windows', node='DESKTOP-CH40V79', release='10', version='10.0.17763', machine='AMD64', processor='Int
el64 Family 6 Model 78 Stepping 3, GenuineIntel')
```

37-getpass module

getpass() prompts the user for a password without echoing. The getpass module provides a secure way to handle the password prompts where programs interact with the users via the terminal.

getuser() function displays the login name of the user. This function checks the environment variables LOGNAME, USER, LNAME and USERNAME, in order, and returns the value of the first non-empty string.

getpass()

To handle passwords

In this program, I need to read a password

Vi read_pass.py

```
import getpass
db_pass=getpass.getpass()
print(f"The entered passwd is: {db_pass}")
```

```
C:\Users\Automation\Desktop\PythonScripts>python read_pass.py
Password:
The entered passwd is: db3456
```

```
import getpass
db_pass=getpass.getpass(prompt="Enter your db pass: ")
print(f"The entered passwd is: {db_pass}")
```

```
C:\Users\Automation\Desktop\PythonScripts>python read_pass.py
Enter your db pass:
```

getuser()

To handle users environment variables

```
>>> import getpass
>>> getpass.getuser()
'ec2-user'
```

```
[ec2-user@ip-172-31-93-165 Udemy]$ env | grep ec2-user
USER=ec2-user
MAIL=/var/spool/mail/ec2-user
PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/opt/aws/bin:/home/ec2-user/.local/bin:/home/ec2-user/bi
n
PWD=/home/ec2-user/Udemy
HOME=/home/ec2-user
LOGNAME=ec2-user
OLDPWD=/home/ec2-user
[ec2-user@ip-172-31-93-165 Udemy]$
```

9-sys module:

(The sys module is used to work with python runtime environment)

Note:

Sys means, it is helpful to get runtime environment on your system.

Introduction to sys module:

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment.

How to use sys module ?

import sys

How to get help on sys module ?

dir(sys), help(sys)

38-Introduction to sys module

```
import sys
#print(sys.version)
#print(sys.version_info)
print(sys.modules)
```

```
print(sys.path)
```

```
['F:\\Udemy videos\\Complete Python Scripting\\Modules\\sys', 'C:\\Python37\\python37.zip', 'C:\\Python37\\DLLs',
'C:\\Python37\\lib', 'C:\\Python37', 'C:\\Python37\\lib\\site-packages']
[Finished in 0.9s]
```

```
import sys
#print(sys.version)
#print(sys.version_info)
#print(sys.modules)
#print(sys.path)  #sys.path is an environment variable for python
sys.exit()
print("hello")
```

```
sys.exit(2)
```

39-sys.argv _ working with command line arguments with an example

(sys.argv returns a list of command line arguments passed to a Python script.)

```
F:\Udemy videos\Complete Python Scripting\Modules\sys>type commad_line_arguments.py
import sys
print(sys.argv)
F:\Udemy videos\Complete Python Scripting\Modules\sys>python commad_line_arguments.py
['commad_line_arguments.py']

F:\Udemy videos\Complete Python Scripting\Modules\sys>python commad_line_arguments.py python shell perl
['commad_line_arguments.py', 'python', 'shell', 'perl']

F:\Udemy videos\Complete Python Scripting\Modules\sys>
```

```
import sys
print(sys.argv)
print(sys.argv[0])
```

```
print(sys.argv[0])
```

```
F:\Udemy videos\Complete Python Scripting\Modules\sys>python commad_line_arguments.py python
['commad_line_arguments.py', 'python']
commad_line_arguments.py
```

```
usr_str=input("Enter your string: ")
usr_action=input("Enter your action on your string (valid actions are: lower/upper/title): ")

if usr_action=="lower":
        print(usr_str.lower())
elif usr_action=="upper":
        print(usr_str.upper())
elif usr_action=="title":
        print(usr_str.title())
else:
        print("Your option is invalid. Please select valida option from this list: lower/upper/title")
```

```
import sys
usr_str=sys.argv[1]
usr_action=sys.argv[2]

if usr_action=="lower":
    print(usr_str.lower())
elif usr_action=="upper":
    print(usr_str.upper())
elif usr_action=="title":
    print(usr_str.title())
else:
    print("Your option is invalid. Please select valida option from this list: lower/upper/title")
```

```
F:\Udemy videos\Complete Python Scripting\Modules\sys>python action_on_strings.py  "python ScripTING"  lower
python scripting

F:\Udemy videos\Complete Python Scripting\Modules\sys>python action_on_strings.py  "python ScripTING"  upper
PYTHON SCRIPTING

F:\Udemy videos\Complete Python Scripting\Modules\sys>python action_on_strings.py  "python ScripTING"  title
Python Scripting

F:\Udemy videos\Complete Python Scripting\Modules\sys>
```

```python
import sys

if len(sys.argv) != 3:
    sys.exit()

usr_str=sys.argv[1]
usr_action=sys.argv[2]

if usr_action=="lower":
    print(usr_str.lower())
elif usr_action=="upper":
    print(usr_str.upper())
elif usr_action=="title":
    print(usr_str.title())
else:
    print("Your option is invalid. Please select valida option from this list: lower/upper/title")
```

```python
import sys

if len(sys.argv) != 3:
    print("Usage:")
    print(f'{sys.argv[0]} <your_req_string>  <lower|upper|title> )
    sys.exit()

usr_str=sys.argv[1]
usr_action=sys.argv[2]

if usr_action=="lower":
    print(usr_str.lower())
elif usr_action=="upper":
    print(usr_str.upper())
elif usr_action=="title":
    print(usr_str.title())
else:
    print("Your option is invalid. Please select valida option from this list: lower/upper/title")
```

```
F:\Udemy videos\Complete Python Scripting\Modules\sys>python action_on_strings.py "python scirpNG"
Usage:
action_on_strings.py <your_req_string>  <lower|upper|title>

F:\Udemy videos\Complete Python Scripting\Modules\sys>
```

10-OS Module(file & dir):

40-Introduction to OS Module and Basic operation

os.sep
os.getcwd()
os.chdir(path)
os.listdir()
os.listdir(path)
os.mkdir(path)
os.makedirs(path) (Recursive directory creation function.)
os.remove(path)
os.removedirs(path) (Remove directories recursively)
os.rmdir(path)
os.rename(src, dst)
os.environ()
os.getuid()
os.getpid()

41-os.path module

os.path is a sub module of os

```
>>> dir(os.path)
['__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_get_sep', '_joinrealpath', '_varprog', '_varprogb', 'abspath', 'altsep', 'basename', 'commonpath', 'commonprefix', 'curdir', 'defpath', 'devnull', 'dirname', 'exists', 'expanduser', 'expandvars', 'extsep', 'genericpath', 'getatime', 'getctime', 'getmtime', 'getsize', 'isabs', 'isdir', 'isfile', 'islink', 'ismount', 'join', 'lexists', 'normcase', 'normpath', 'os', 'pardir', 'pathsep', 'realpath', 'relpath', 'samefile', 'sameopenfile', 'samestat', 'sep', 'split', 'splitdrive', 'splitext', 'stat', 'supports_unicode_filenames', 'sys']
```

(os.path module is used to work on paths)

os.path

os.path.sep
os.path.basename(path)
os.path.dirname(path)
os.path.join(path1,path2)
os.path.split(path) --> is used to Split the path name into a pair head and tail.
os.path.getsize(path) --> in bytes
os.path.exists(path)
os.path.isfile(path)
os.path.isdir(path)
os.path.islink(path)

After time module we will also discuss: getatime, getctime and getmtime

```
import os
path="/home/ec2-user/test.py"
print(os.path.basename(path))
print(os.path.dirname(path))
```

```
[ec2-user@ip-172-31-93-165 Udemy]$ cat os_path.py
import os
path="/home/ec2-user/test.py"
print(os.path.basename(path))
print(os.path.dirname(path))
[ec2-user@ip-172-31-93-165 Udemy]$ python3 os_path.py
test.py
/home/ec2-user
[ec2-user@ip-172-31-93-165 Udemy]$
```

It is helpful, when you're installing something.

```
>>> path1="/home"
>>> path2="ec2-user"
>>> print(path1+path2)
/homeec2-user
>>> print(path1+'/'+path2)
/home/ec2-user
>>> import os
>>> print(os.path.join(path1,path2))
/home/ec2-user
>>>
```

```
/home/ec2-user
>>> path="/home/ec2-user/test.py"
>>> print(os.path.split(path))
('/home/ec2-user', 'test.py')
>>>
```

```
import os
path="/home/ec2-user/test.py"
print(os.path.basename(path))
print(os.path.dirname(path))

if os.path.exists(path):
  print("your file is tehre")
else:
  print(f'{path} is not present on this host')
```

```
[ec2-user@ip-172-31-93-165 Udemy]$ cat os_path.py
import os
path="/home/ec2-user/test.py"
'''
print(os.path.basename(path))
print(os.path.dirname(path))

if os.path.exists(path):
  print("your file is tehre")
else:
  print(f'{path} is not present on this host')
'''

if os.path.isfile(path):
   print(f"{path} is a file")
else:
   print(f"{path} is a dir")
[ec2-user@ip-172-31-93-165 Udemy]$ python3 os_path.py
/home/ec2-user/test.py is a file
```

42-os.system() function from os module

(os.system() is used to execute os commands)

Os.sytstem() will execute os commands and should return “0” for success. Unless “1..etc” for unsuccess

```
>>> import os
>>> os.getcwd()
'/home/ec2-user/Udemy'
>>> os.listdir()
['allign_text.py', 'xyz', 'narendra', 'read_pass.py', 'os_path.py', 'myos']
>>> os.system("pwd")
/home/ec2-user/Udemy
0
>>> os.system("ls")
allign_text.py  myos  narendra  os_path.py  read_pass.py  xyz
0
```

```
>>> os.system("pwd")
/home/ec2-user/Udemy
0
>>> os.system("adf")
sh: adf: command not found
32512
>>> rt=os.system("ls")
allign_text.py  myos  narendra  os_path.py  read_pass.py  xyz
>>>
```

43-Practice script on platform and os module

(write a single python script to clear terminal of any Operating system)

Or

(Write a platform independent script to clear terminal)

```
[ec2-user@ip-172-31-93-165 Udemy]$ cat clear_screen.py
import os
import platform
if platform.system()=="Windows":
    os.system("cls")
else:
    os.system("clear")

[ec2-user@ip-172-31-93-165 Udemy]$
```

44-os.walk(path)

used* to generate directory and file names *in a* directory tree *by walking

Note: First complete for loop then start this os.walk

```
find path -name *.txt

/home/xyz
/home/xyz/dir1
/home/xyz/dir1/sub1
```

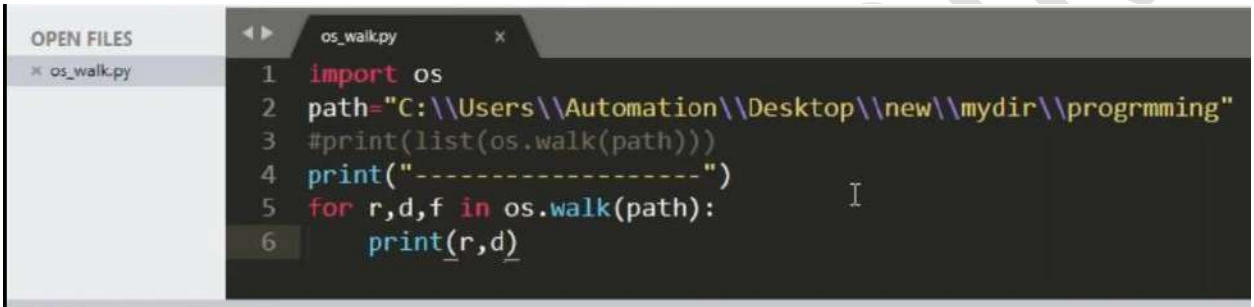
```
os_walk.py

1 import os
2 path="C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming"
3 print(list(os.walk(path)))

[('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming', ['hello'], ['ok.txt', 'x.txt', 'y.py']),
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming\\hello', [], [])]
[Finished in 1.0s]
```

```
OPEN FILES
os_walk.py

1 import os
2 path="C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming"
3 print(list(os.walk(path)))
4 print("--------------------")
5 for each in list(os.walk(path)):
6     print(each)

[('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming', ['byee', 'hello'], ['bye', 'ok.txt', 'x.txt', 'y.py']),
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming\\byee', [], []),
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming\\hello', ['helo1'], ['hello.py', 'hello.txt']),
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming\\hello\\helo1', [], [])]
--------------------
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming', ['byee', 'hello'], ['bye', 'ok.txt', 'x.txt', 'y.py'])
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming\\byee', [], [])
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming\\hello', ['helo1'], ['hello.py', 'hello.txt'])
('C:\\Users\\Automation\\Desktop\\new\\mydir\\progrmming\\hello\\helo1', [], [])
[Finished in 1.0s]
```

Unpack your tuple

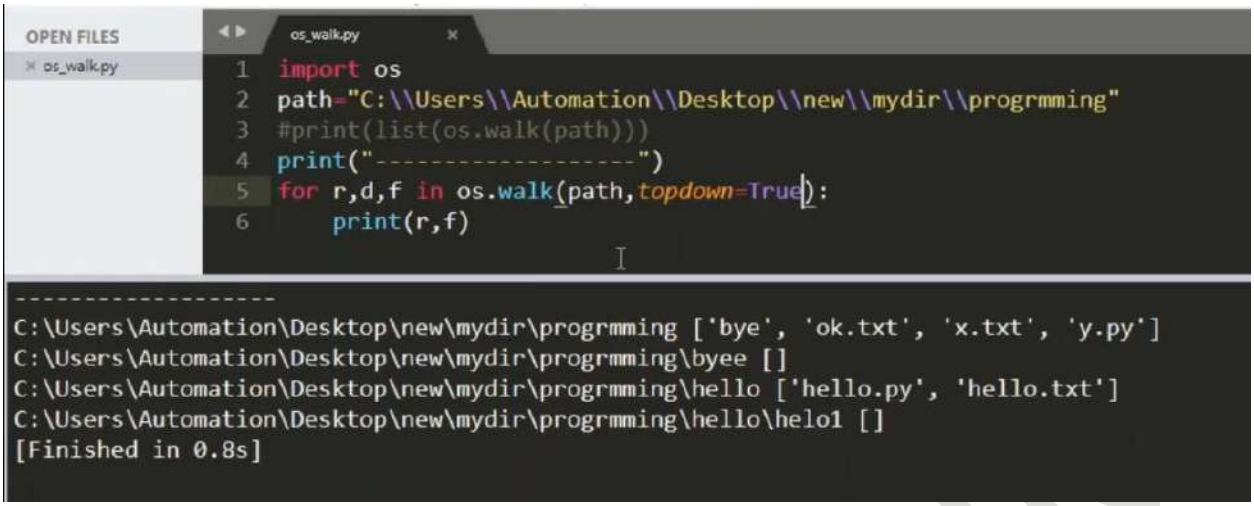
```
>>> x=(3,4,5)
>>> x
(3, 4, 5)
>>> x,y,z=(3,4,5)
>>> y
4
>>> z
5
>>> x
3
>>>
```


```
--------------------
C:\Users\Automation\Desktop\new\mydir\progrmming
C:\Users\Automation\Desktop\new\mydir\progrmming\byee
C:\Users\Automation\Desktop\new\mydir\progrmming\hello
C:\Users\Automation\Desktop\new\mydir\progrmming\hello\helo1
[Finished in 0.8s]
```

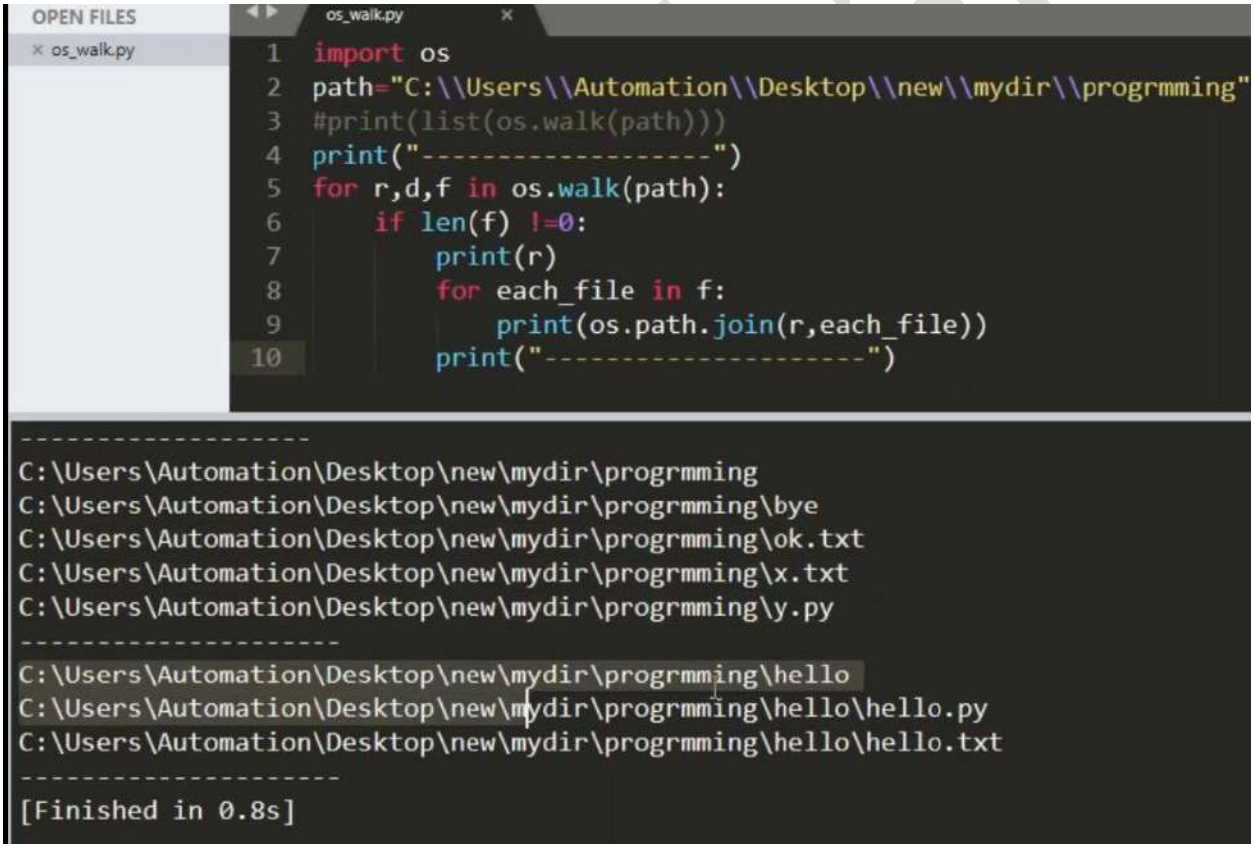



```
--------------------
C:\Users\Automation\Desktop\new\mydir\progrmming ['byee', 'hello']
C:\Users\Automation\Desktop\new\mydir\progrmming\byee []
C:\Users\Automation\Desktop\new\mydir\progrmming\hello ['helo1']
C:\Users\Automation\Desktop\new\mydir\progrmming\hello\helo1 []
[Finished in 0.8s]
```


```
--------------------
C:\Users\Automation\Desktop\new\mydir\progrmming ['bye', 'ok.txt', 'x.txt', 'y.py']
C:\Users\Automation\Desktop\new\mydir\progrmming\byee []
C:\Users\Automation\Desktop\new\mydir\progrmming\hello ['hello.py', 'hello.txt']
C:\Users\Automation\Desktop\new\mydir\progrmming\hello\helo1 []
[Finished in 0.7s]
```



By default topdown=True



45-Best Practice with os.walk for real-time

```
import os
req_file=input("Enter your file name to search: ")
for r,d,f in os.walk("/home/ec2-user"):
    for each_file in f:
      if each_file==req_file
      print(os.path.join(r,each_file))
```

```
import os

req_file=input("Enter your file name to search: ")

for r,d,f in os.walk("C:\\Users\\Automation\\Desktop"):
    for each_file in f:
        if each_file==req_file:
            print(os.path.join(r,each_file))
```

os_walk.py
search_a_file.py

```
import os
import string
pd_names=string.ascii_uppercase
vd_names=[]
for each_drive in pd_names:
    if os.path.exists(each_drive+":\\"):
        print(each_drive)
        vd_names.append(each_drive+":\\")
print(vd_names)

'''
```

```
C
E
F
['C:\\', 'E:\\', 'F:\\']
[Finished in 0.9s]
```

11-Loops - for and while loops with break, continue and pass:

46-Practice - Read a path and check if given path is a file or a directory

```python
import os
path=input("Enter your path: ")
if os.path.isfile(path):
    print(f"The given path: {path} is a file")
else:
    print(f"The given path: {path} is a directory")
```

```
C:\Users\Automation\Desktop\PythonScripts>python path_file_dir.py
Enter your path: C:\\Users\\Automation\\Desktop\\PythonScripts
The given path: C:\\Users\\Automation\\Desktop\\PythonScripts is a directory

C:\Users\Automation\Desktop\PythonScripts>python path_file_dir.py
Enter your path: C:\\Users\\Automation\\Desktop\\PythonScripts\\path_file_dir.py
The given path: C:\\Users\\Automation\\Desktop\\PythonScripts\\path_file_dir.py is a file
```

```
C:\Users\Automation\Desktop\PythonScripts>python path_file_dir.py
Enter your path: x:\\python
The given path: x:\\python is a directory
```

```python
import os
path=input("Enter your path: ")

if os.path.exists(path):
    print(f"Given path : {path}is a valid path")
else:
    print(f"Given path : {path} is not existing on this host")
```

```
C:\Users\Automation\Desktop\PythonScripts>python path_file_dir.py
Enter your path: C:\\Users\\Automation\\Desktop\\PythonScripts
Given path : C:\\Users\\Automation\\Desktop\\PythonScriptsis a valid path

C:\Users\Automation\Desktop\PythonScripts>
```

```python
import os
path=input("Enter your path: ")

if os.path.exists(path):
    print(f"Given path : {path}is a valid path")
    if os.path.isfile(path):
        print("and it is a file path")
    else:
        print("and it is a directory path")
else:
    print(f"Given path : {path} is not existing on this host")
```

```
C:\Users\Automation\Desktop\PythonScripts>python path_file_dir.py
Enter your path: C:\\Users\\Automation\\Desktop\\PythonScripts\\path_file_dir.py
Given path : C:\\Users\\Automation\\Desktop\\PythonScripts\\path_file_dir.pyis a valid path
and it is a file path
```

47-Introduction to loops with an example

```python
import os
path=input("Enter your directory path: ")
df_l=os.listdir(path)
#print(df_l)
p1=df_l[0]
p2=df_l[1]

if os.path.isfile(p2):
    print(f"{p2} is a file")
else:
    print(f"{p2} is a directory")
```

Complete path for your file or directory

```python
p2=os.path.join(path,df_l[1])
```

```
C:\Users\Automation\Desktop\PythonScripts>python read_a_path_and_identify_files_and_dirs.py
Enter your directory path: C:\\Users\\Automation\\Desktop\\new
C:\\Users\\Automation\\Desktop\\new\xyz.sh
```

```python
import os
path=input("Enter your directory path: ")
df_l=os.listdir(path)
#print(df_l)
p1=os.path.join(path,df_l[0])
p2=os.path.join(path,df_l[1])

if os.path.isfile(p1):
    print(f"{p1} is a file")
else:
    print(f"{p1} is a directory")

if os.path.isfile(p2):
    print(f"{p2} is a file")
else:
    print(f"{p2} is a directory")
```

```python
import os
import sys
path=input("Enter your directory path: ")
if os.path.exists(path):
    df_l=os.listdir(path)
else:
    print("please provide valid path")
    sys.exit()



#print(df_l)
p1=os.path.join(path,df_l[0])
p2=os.path.join(path,df_l[1])

if os.path.isfile(p1):
    print(f"{p1} is a file")
else:
    print(f"{p1} is a directory")
```

For loop

Syntax: *For <variable> in [list] or (tuple) or "string"*

```python
for each in "narendra":
    print("ok")
    print("hi")
```

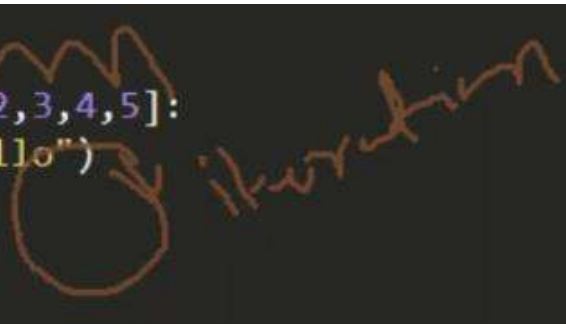
```
ok
hi
ok
hi
[Finished in 0.6s]
```

```python
for each in [2,3,4,5]:
    print("hello")
```

```
hello
hello
hello
hello
[Finished in 0.9s]
```

Loop iterations

```
for each in [2,3,4,5]:
    print("hello")
```



```
print("befor loop")

for each in [2,3,4,5]:
    print("hello",each)


print("after loop")
```

```
befor loop
hello 2
hello 3
hello 4
hello 5
after loop
[Finished in 1.8s]
```

```
import os
import sys
path=input("Enter your directory path: ")
if os.path.exists(path):
    df_l=os.listdir(path)
else:
    print("please provide valid path")
    sys.exit()


list_of_files_dir=os.listdir(path)
print("all files and dirs: ",list_of_files_dir)
for each_file_or_dir in list_of_files_dir:
    f_d_p=os.path.join(path,each_file_or_dir)
    print(f_d_p)
```

```
C:\Users\Automation\Desktop\PythonScripts>python read_a_path_and_identify_files_and_dirs.py
Enter your directory path: C:\\Users\\Automation\\Desktop\\new
all files and dirs:  ['mydir', 'x.txt', 'y.py', 'z.sh']
C:\\Users\\Automation\\Desktop\\new\mydir
C:\\Users\\Automation\\Desktop\\new\x.txt
C:\\Users\\Automation\\Desktop\\new\y.py
C:\\Users\\Automation\\Desktop\\new\z.sh
```

If else block inside for loop

```
list_of_files_dir=os.listdir(path)
print("all files and dirs: ",list_of_files_dir)
for each_file_or_dir in list_of_files_dir:
    f_d_p=os.path.join(path,each_file_or_dir)
    if os.path.isfile(f_d_p):
        print(f'{f_d_p} is a file')
    else:
        print(f'{f_d_p} is a directory')
```

48-Loops _ Working with for loop

Loops are to repeat a block of code multiple times.

```
for each_value in [4,5,6]:
    print('---------')
    print("This is a iteration")
    print("_________")
```

```
---------
This is a iteration
_________
---------
This is a iteration
_________
---------
This is a iteration
_________
[Finished in 0.5s]
```

```
for each_char in "python":
    print("---------",each_char)
```

```
--------- p
--------- y
--------- t
--------- h
--------- o
--------- n
[Finished in 0.6s]
```

```
for each_char in "python":
    print("---------",each_char)
print("we are working with loops")
```

```
--------- p
--------- y
--------- t
--------- h
--------- o
--------- n
we are working with loops
[Finished in 0.5s]
```

```
my_list=[3,4,34,5,67,89,23]

for each in my_list:
    rem=each%2
    if rem==0:
        print(f"{each} is even")
    else:
        print(f"{each} is odd")
```

```
3 is odd
4 is even
34 is even
5 is odd
67 is odd
89 is odd
23 is odd
[Finished in 0.8s]
```

49-Simple practice with for loop

```
usr_in = input("Please Enter your Input string:")
i = 0
for listv in usr_in:
    print(f"{listv}-->{i}")
    i = i+1
```

Output:

C:\Py_Projects\Invoice_Billing>python forloop.py

p-->0

y-->1

t-->2

h-->3

o-->4

n-->5

50-Find all files in a directory with required extension.py_.sh_.log_.txt etc

```
#!/usr/local/bin/python3
import os
req_path=input("Enter your directory path: ")
#req_ex=input("Enter the required files extention .py/.sh/.log/.txt: ")

if os.path.isfile(req_path):
  print(f"The given path {req_path} is a file. Please pass only directory path")
else:
  all_f_ds=os.listdir(req_path)
  if len(all_f_ds)==0:
     print(f"The given path is {req_path} an empty path")
  else:
     req_ex=input("Enter the required files extention .py/.sh/.log/.txt: ")
     req_files=[]
     for each_f in all_f_ds:
        if each_f.endswith(req_ex):
           req_files.append(each_f)
     if len(req_files)==0:
        print(f"There are no {req_ex} files in the logcation of {req_path}")
     else:
        print(f"There are {len(req_files)} files in the location of {req_path} with an extention of {req_ex}")
        print(f"The required files are: {req_files}")
```

```
[ec2-user@ip-172-31-93-165 ~]$ ./get_req_ext_files.py
Enter your directory path: /home/ec2-user/Udemy
Enter the required files extention .py/.sh/.log/.txt: .py
There are 5 files in the location of /home/ec2-user/Udemy with an extention of .py
So, the files are: ['allign_text.py', 'clear_screen.py', 'run_os_cmds.py', 'read_pass.py', 'os_
[ec2-user@ip-172-31-93-165 ~]$ ./get_req_ext_files.py
Enter your directory path: /home/ec2-user/Udemy
Enter the required files extention .py/.sh/.log/.txt: .sh
There are 2 files in the location of /home/ec2-user/Udemy with an extention of .sh
So, the files are: ['start_httpd.sh', 'hello.sh']
[ec2-user@ip-172-31-93-165 ~]$
```

51-Complete range() function

Which is use to generate only integers as a list

```
>>> print range(5)
[0, 1, 2, 3, 4]
>>> print rang(3)
[0, 1, 2]
```

```
C:\Users\Automation>c:\Python27\python.exe
Python 2.7.16 (v2.7.16:413a49145e, Mar  4 2
Type "help", "copyright", "credits" or "lic
>>> print range(5)
[0, 1, 2, 3, 4]
>>> print range(3)
[0, 1, 2]
>>> exit()

C:\Users\Automation>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64
Type "help", "copyright", "credits" or "license" for more information.
>>> print(ragne(5))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'ragne' is not defined
>>> print(range(5))
range(0, 5)
>>> print(range(3))
range(0, 3)
>>> print(list(range(5)))
```

syntax:
range(start,stop,step)
3-argument ==> By default start=0, step=1

it will generate list

```
>>> print(list(range(0,5)))
[0, 1, 2, 3, 4]
>>> print(list(range(0,20,1)))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> print(list(range(0,20,2)))
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>> print(list(range(10,2)))
[]
>>> print(list(range(10,2,-1)))
[10, 9, 8, 7, 6, 5, 4, 3]
```

in python3 it will generate

```
>>> print(list(range(5)))
[0, 1, 2, 3, 4]
>>>
```

But you can convert your range object to a list with ex:

```
>>> print(list(range(10,2)))
[]
>>> print(list(range(10,2,-1)))
[10, 9, 8, 7, 6, 5, 4, 3]
>>> print(list(range(-2,-10,-3)))
[-2, -5, -8]
>>>
```

```python
my_list=[5,6,7,34,"Python"]

#print(list(range(len(my_list))))

#print(my_list[0])
for each_index in range(len(my_list)):
    print(f'Index-->{each_index}: value-->{my_list[each_index]}')
```

```
Index-->0: value-->5
Index-->1: value-->6
Index-->2: value-->7
Index-->3: value-->34
Index-->4: value-->Python
[Finished in 0.8s]
```

52-for loop to work with strings, list, tuple and dictionaries

Strings

```
my_string="working with for loop"
print(my_string)

#print("\n".join(my_string))
'''
for each_char in my_string:
    print(each_char)
'''
```

List

```
my_list=[1,2,3,4,5]
for each_value in my_list:
    print(each_value)
```

Tuple

```
my_list=[(1,2),(4,5),(6,7)]
for each_item in my_list:
    print(each_item)
```

```
working with for loop
(1, 2)
(4, 5)
(6, 7)
[Finished in 0.6s]
```

```
my_list=[(1,2),(4,5),(6,7)]
for f,s in my_list:
    print(s)
```

```
working with for loop
2
5
7
[Finished in 0.7s]
```

Dictionary

```
my_dic={'a':1,'b':2,'c':3}
for each in my_dic:
    print(each)
```

```
working with for loop
a
b
c
[Finished in 0.7s]
```

```
my_dic={'a':1,'b':2,'c':3}
for each in my_dic.items():
    print(each)
```

```
working with for loop
('a', 1)
('b', 2)
('c', 3)
[Finished in 0.7s]
```

```
my_dic={'a':1,'b':2,'c':3}
for key,value in my_dic.items():
    print(key,value)
```

```
working with for loop
a 1
b 2
c 3
[Finished in 0.7s]
```

```
for each in my_dic.keys():
    print(each)
```

```
for each in my_dic.values():
    print(each)
```

Suppose we have the days and temperatures in two lists as shown below.

```
days = ["Sunday",
"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday"]
temp_C = [30.5,32.6,31.8,33.4,29.8,30.2,29.9]
```

We can now proceed to use dictionary comprehension to create a dictionary of weekly temperatures.

```
# Creating a dictionary of weekly tempertaures
# from the list of temperatures and days
weekly_temp = {day:temp for (day,temp) in zip(days,temp_C)}
print(weekly_temp)
# Output
{'Sunday': 30.5, 'Monday': 32.6, 'Tuesday': 31.8, 'Wednesday': 33.4,
'Thursday': 29.8, 'Friday': 30.2, 'Saturday': 29.9}
```

https://www-freecodecamp-org.cdn.ampproject.org/c/s/www.freecodecamp.org/news/dictionary-comprehension-in-python-explained-with-examples/amp/

53-Introduction to while loop

while loop:

while loop is used to execute a block of statements repeatedly until a given a condition is satisfied.

```
while True:
    print("Welcome to loops")
    print(".....")
```

While Is used to repeat your logic infinite no.of times.

If you want to monitor your file system infinite no.of times on every 1 sec

```
import time
while True:
    print("MOnitoring file system usage")
    time.sleep(1)
```

Condition based

```
value=4
while value<=6789:
    print(value)
    value=value+456
```

```
4
460
916
1372
1828
2284
2740
3196
3652
4108
4564
5020
5476
5932
6388
[Finished in 0.3s]
```

```
cnt=1
while cnt <=5:
    print("hello")
    cnt=cnt+1
```

```
hello
hello
hello
hello
hello
[Finished in 0.2s]
```

54-Loop Control statements - break, continue and pass

Control statements:

- break
- continue

Break: Is to stop the loop
Continue: is to continue of the loop
Pass: it will not do anything

break

```
import sys
print("before exit")
sys.exit()
print("after exit")
for each in [3,4,56,7,8]:
    break
    print(each)

print("after loop")
```

Note: stopping script is different, and stopping loop is different.

```
for each in [3,4,56,7,8]:
    break
    print(each)

print("after loop")
```

```
after loop
[Finished in 0.2s]
```

```
for each in [3,4,56,7,8]:
    print(each)
    break

print("after loop")
```

```
3
after loop
[Finished in 0.2s]
```

```
for each in [3,4,56,7,8]:
    print(each)
    if each==56:
        break

print("after loop")
```

```
3
4
56
after loop
[Finished in 0.2s]
```

```
11  paths=['/usr/bin','/usr/bin/httpd','/home/users/xyz/weblogic/config.xml']
12  for each_path in paths:
13      print("now working on: ",each_path)
14      if 'httpd' in each_path:
15          print(each_path)
16          break
17  print("outside of for loop")
18
19
20
```

```
now working on:  /usr/bin
now working on:  /usr/bin/httpd
/usr/bin/httpd
outside of for loop
[Finished in 0.2s]
```

```
cnt=1
while True:
    print(cnt)
    if cnt==100:
        break
    cnt=cnt+1
```

Continue

```
for each in range(1,11):
    if each ==7:
        continue
    print(each)
```

```
1
2
3
4
5
6
8
9
10
[Finished in 0.1s]
```

```
for each in range(1,11):
    if each ==7:
        continue
        print("this is the line inside of your if condition after continue keyword")
    print(each)
```

```
1
2
3
4
5
6
8
9
10
[Finished in 0.1s]
```

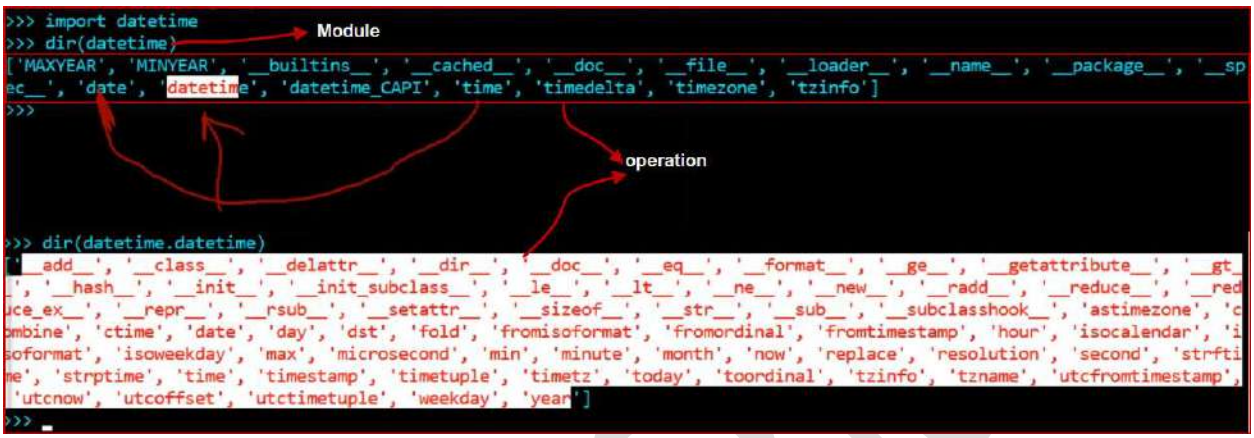
When ever we use continue with condition, that condition value will not execute and it will continue after the condition.

Pass

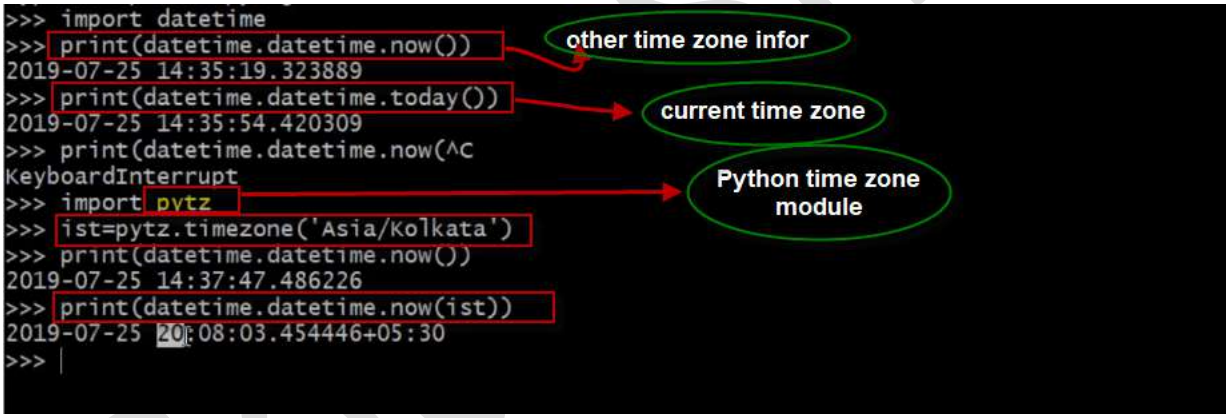
```
if True:
    pass
```

12-datetime module to work with dates and times:

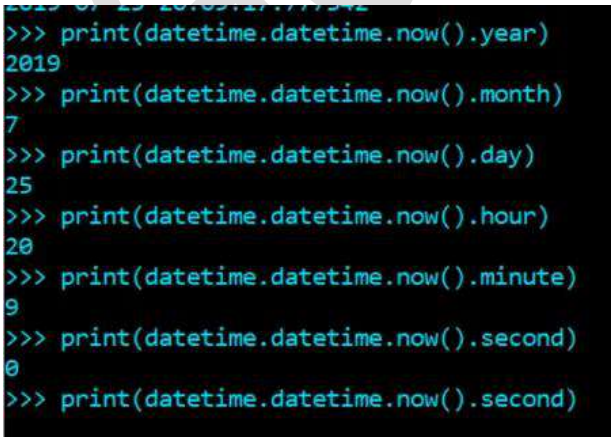
55-Introduction to datetime module



timezones



Year month day hour min sec



Format year month day

```
>>> print(datetime.datetime.now().strftime("%y-%m-%d")
... )
19-07-25
>>> print(datetime.datetime.now().strftime("%Y-%m-%d"))
2019-07-25
```

Strftime.org

Python `strftime` cheatsheet

https://strftime.org/

CodeExample	Description
%aSun	Weekday as locale's abbreviated name.
%ASunday	Weekday as locale's full name.
%w0	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.
%d08	Day of the month as a zero-padded decimal number.
%-d8	Day of the month as a decimal number. (Platform specific)
%bSep	Month as locale's abbreviated name.
%BSeptember	Month as locale's full name.
%m09	Month as a zero-padded decimal number.
%-m9	Month as a decimal number. (Platform specific)
%y13	Year without century as a zero-padded decimal number.
%Y2013	Year with century as a decimal number.
%H07	Hour (24-hour clock) as a zero-padded decimal number.
%-H7	Hour (24-hour clock) as a decimal number. (Platform specific)
%I07	Hour (12-hour clock) as a zero-padded decimal number.
%-I7	Hour (12-hour clock) as a decimal number. (Platform specific)
%pAM	Locale's equivalent of either AM or PM.
%M06	Minute as a zero-padded decimal number.
%-M6	Minute as a decimal number. (Platform specific)
%S05	Second as a zero-padded decimal number.
%-S5	Second as a decimal number. (Platform specific)
%f000000	Microsecond as a decimal number, zero-padded on the left.

CodeExample	Description
%z+0000	UTC offset in the form ±HHMM[SS[.ffffff]] (empty string if the object is naive).
%ZUTC	Time zone name (empty string if the object is naive).
%j251	Day of the year as a zero-padded decimal number.
%-j251	Day of the year as a decimal number. (Platform specific)
%U36	Week number of the year (Sunday as the first day of the week) as a zero padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.
%W35	Week number of the year (Monday as the first day of the week) as a decimal number. All days in a new year preceding the first Monday are considered to be in week 0.
%cSun Sep 8 07:06:05 2013	Locale's appropriate date and time representation.
%x09/08/13	Locale's appropriate date representation.
%X07:06:05	Locale's appropriate time representation.
%%%	A literal '%' character.

Convert seconds to date

```
>>> print(datetime.datetime.fromtimestamp(1555555555))
2019-04-18 08:15:55
```

56-Practice - Find the files which are older than x days from a given path

```
import os
import sys
req_path=input("Enter your path: ")
age=3
if not os.path.exists(req_path):
   print("Please provide valid path ")
   sys.exit(1)
if os.path.isfile(req_path):
   print("Please provide directory path ")
   sys.exit(2)

for each_file in os.listdir(req_path):
   each_file_path=os.path.join(req_path,each_file)
   if os.path.isfile(each_file_path):
      print(each_file_path)
```

```
>>> dir(os.path)
['__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_get_sep', '_joinrealpath', '_varprog', '_varprogb', 'abspath', 'altsep', 'basename', 'commonpath', 'commonprefix', 'curdir', 'defpath', 'devnull', 'dirname', 'exists', 'expanduser', 'expandvars', 'extsep', 'genericpath', 'getatime', 'getctime', 'getmtime', 'getsize', 'isabs', 'isdir', 'isfile', 'islink', 'ismount', 'join', 'lexists', 'normcase', 'normpath', 'os', 'pardir', 'pathsep', 'realpath', 'relpath', 'samefile', 'sameopenfile', 'samestat', 'sep', 'split', 'splitdrive', 'splitext', 'stat', 'supports_unicode_filenames', 'sys']
```

```
>>> print(os.path.getctime("/home/ec2-user/Udemy/tomcat.log"))
1563974954.165263
```

```
>>> import datetime
>>> datetime.datetime.fromtimestamp(os.path.getctime("/home/ec2-user/Udemy/tomcat.log"))
datetime.datetime(2019, 7, 24, 13, 29, 14, 165263)
```

```
import os
import sys
import datetime
req_path=input("Enter your path: ")
age=3
if not os.path.exists(req_path):
   print("Please provide valid path ")
   sys.exit(1)
if os.path.isfile(req_path):
   print("Please provide directory path ")
   sys.exit(2)

for each_file in os.listdir(req_path):
   each_file_path=os.path.join(req_path,each_file)
   if os.path.isfile(each_file_path):
      print(each_file_path,datetime.datetime.fromtimestamp(os.path.getctime(each_file_path)))
```

```
[ec2-user@ip-172-31-93-165 ~]$ python3 delete_older_than_x_day.py
Enter your path: /home/ec2-user/Udemy/
/home/ec2-user/Udemy/y.txt 2019-07-24 13:29:14.165263
/home/ec2-user/Udemy/allign_text.py 2019-07-20 11:13:03.799637
/home/ec2-user/Udemy/start_httpd.sh 2019-07-24 13:29:23.877273
/home/ec2-user/Udemy/a.out 2019-07-21 15:51:47.486365
```

```
  if os.path.isfile(each_file_path):
     file_cre_date=datetime.datetime.fromtimestamp(os.path.getctime(each_file_path))
     print(each_file_path,today-file_cre_date)
```

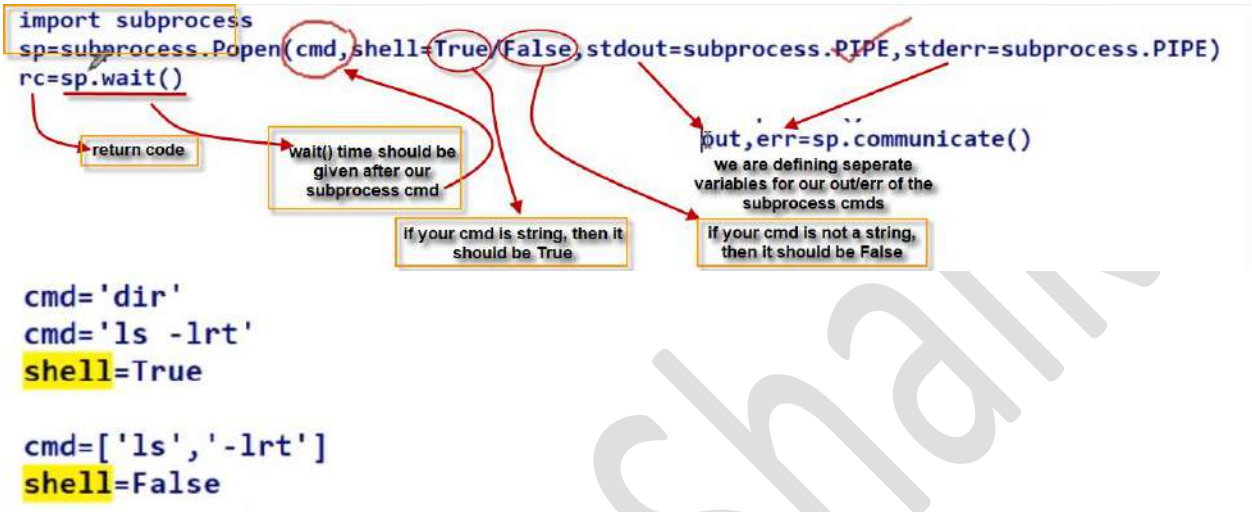
```
[ec2-user@ip-172-31-93-165 ~]$ python3 delete_older_than_x_day.py
Enter your path: /home/ec2-user/Udemy
/home/ec2-user/Udemy/y.txt 1 day, 2:31:58.840586
/home/ec2-user/Udemy/allign_text.py 5 days, 4:48:09.206212
/home/ec2-user/Udemy/start_httpd.sh 1 day, 2:31:49.128576
```

13-subprocess Module - To execute any Operating System Commands with python:

Os.system("ls")

Os.system("dir")

Out= Os.system("dir") → here os.system will directly run your command, and status will not stored into a variable.
Print(out)
And will return only in Boolean values either "0" or "1"
So to overcome this issue and to store the result in to a variable, we will use "subprocess" module



Import subprocess
Sp = subprocess.Popen(cmd,shell=True/False, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
Rc=sp.wait()
Out,err=sp.communicate()

If the cmd is True

```
import subprocess
cmd="ls -lrt"
sp=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE)
rc=sp.wait()
out,err=sp.communicate()

print(f'The retrun code: {rc}')
print(f'The output :\n{out}')
print(f'The error : \n{err}')
```

Output with b' (is a byte code, python is treating as a byte object)

```
[ec2-user@ip-172-31-93-165 ~]$ python3 execute_cmds.py
The retrun code: 0
The output :
b'total 44\ndrwxrwxr-x 2 ec2-user ec2-user 4096 Jul  6 05:50 PythonScripts\n-rw-rw-r-- 1 ec
08 test.py\ndrwxrwxr-x 3 ec2-user ec2-user 4096 Jul 12 01:12 June-2019\ndrwxrwxr-x 2 ec2-us
arendra\nlrwxrwxrwx 1 ec2-user ec2-user    7 Jul 21 14:42 hi.txt -> test.py\n-rw-rw-r-- 1 e
:28 tomcat_log.log\n-rw-rw-r-- 1 ec2-user ec2-user    0 Jul 24 13:28 hello.sh\ndrwxrwxr-x 2
13:36 empty\n-rwxrwxr-x 1 ec2-user ec2-user  868 Jul 24 14:01 get_req_ext_files.py\n-rw-rw-
l 25 16:04 delete_older_than_x_day.py\n-rw-r--r-- 1 ec2-user ec2-user 1696 Jul 26 00:55 dow
ec2-user ec2-user  629 Jul 26 01:22 working_with_remote_servers.py\ndrwxrwxr-x 5 ec2-user e
\n-rw-rw-r-- 1 ec2-user ec2-user  245 Jul 28 15:14 execute_cmds.py\n'
The error :
b''
```

Just to avoid that b' ' byte code, to get your output as a string

Add "Universal_newlines=True" in your subprocess command.

```
import subprocess
cmd="ls -lrt"
sp=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_newlines=True)
rc=sp.wait()
out,err=sp.communicate()

print(f'The retrun code: {rc}')
print(f'The output :\n{out}')
print(f'The error : \n{err}')
```

```
[ec2-user@ip-172-31-93-165 ~]$ python3 execute_cmds.py
The retrun code: 0
The output :
total 44
drwxrwxr-x 2 ec2-user ec2-user 4096 Jul  6 05:50 PythonScripts
-rw-rw-r-- 1 ec2-user ec2-user  502 Jul 11 02:08 test.py
drwxrwxr-x 3 ec2-user ec2-user 4096 Jul 12 01:12 June-2019
drwxrwxr-x 2 ec2-user ec2-user 4096 Jul 21 13:39 narendra
lrwxrwxrwx 1 ec2-user ec2-user    7 Jul 21 14:42 hi.txt -> test.py
-rw-rw-r-- 1 ec2-user ec2-user    0 Jul 24 13:28 tomcat_log.log
-rw-rw-r-- 1 ec2-user ec2-user    0 Jul 24 13:28 hello.sh
drwxrwxr-x 2 ec2-user ec2-user 4096 Jul 24 13:36 empty
-rwxrwxr-x 1 ec2-user ec2-user  868 Jul 24 14:01 get_req_ext_files.py
-rw-rw-r-- 1 ec2-user ec2-user  643 Jul 25 16:04 delete_older_than_x_day.py
-rw-r--r-- 1 ec2-user ec2-user 1696 Jul 26 00:55 dowithscripting.pem
-rwxrwxr-x 1 ec2-user ec2-user  629 Jul 26 01:22 working_with_remote_servers.py
drwxrwxr-x 5 ec2-user ec2-user 4096 Jul 27 16:50 Udemy
-rw-rw-r-- 1 ec2-user ec2-user  269 Jul 28 15:15 execute_cmds.py

The error :
```

Output: now you are getting output as a list now with split()

If shell = false, just use split() to get the output in list.

```
import subprocess
cmd="ls -lrt"
sp=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_newlines=True)
rc=sp.wait()
out,err=sp.communicate()

print(f'The retrun code: {rc}')
print(f'The output :\n{out.splitlines()}')
print(f'The error : \n{err.splitlines()}')
```

```
[ec2-user@ip-172-31-93-165 ~]$ python3 execute_cmds.py
The retrun code: 0
The output :
['total 44', 'drwxrwxr-x 2 ec2-user ec2-user 4096 Jul  6 05:50 PythonScripts', '-rw-rw-r-- 1 ec2-user ec2-user  502 Jul 11
 02:08 test.py', 'drwxrwxr-x 3 ec2-user ec2-user 4096 Jul 12 01:12 June-2019', 'drwxrwxr-x 2 ec2-user ec2-user 4096 Jul 21
 13:39 narendra', 'lrwxrwxrwx 1 ec2-user ec2-user    7 Jul 21 14:42 hi.txt -> test.py', '-rw-rw-r-- 1 ec2-user ec2-user
 0 Jul 24 13:28 tomcat_log.log', '-rw-rw-r-- 1 ec2-user ec2-user    0 Jul 24 13:28 hello.sh', 'drwxrwxr-x 2 ec2-user ec2-u
ser 4096 Jul 24 13:36 empty', '-rwxrwxr-x 1 ec2-user ec2-user  868 Jul 24 14:01 get_req_ext_files.py', '-rw-rw-r-- 1 ec2-u
ser ec2-user  643 Jul 25 16:04 delete_older_than_x_day.py', '-rw-r--r-- 1 ec2-user ec2-user 1696 Jul 26 00:55 dowithscript
ing.pem', '-rwxrwxr-x 1 ec2-user ec2-user  629 Jul 26 01:22 working_with_remote_servers.py', 'drwxrwxr-x 5 ec2-user ec2-us
er 4096 Jul 27 16:50 Udemy', '-rw-rw-r-- 1 ec2-user ec2-user  295 Jul 28 15:17 execute_cmds.py']
The error :
[]
```

```
#cmd="ls -lrt".split()
cmd=['ls','-lrt']
sp=subprocess.Popen(cmd,shell=False,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_newlines=True)
rc=sp.wait()
out,err=sp.communicate()

print(f'The retrun code: {rc}')
print(f'The output :\n{out.splitlines()}')
print(f'The error : \n{err.splitlines()}')
```

Note:

if your cmd is like an environment variable, then only you have to take shell=True.

In windows, please make sure your shell=True always

```
cmd='echo $HOME'
sp=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_newlines=True)
rc=sp.wait()
out,err=sp.communicate()

print(f'The retrun code: {rc}')
print(f'The output :\n{out.splitlines()}')
print(f'The error : \n{err.splitlines()}')
'''
```

```
[ec2-user@ip-172-31-93-165 ~]$ python3 execute_cmds.py
The retrun code: 0
The output :
['/home/ec2-user']
The error :
[]
```

57-Introduction to subprocess module

```
[ec2-user@ip-172-31-93-165 ~]$ bash --version
GNU bash, version 4.2.46(2)-release (x86_64-redhat-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[ec2-user@ip-172-31-93-165 ~]$
```

58-Practice-1 with subprocess module

```
[ec2-user@ip-172-31-93-165 ~]$ bash --version
GNU bash, version 4.2.46(2)-release (x86_64-redhat-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[ec2-user@ip-172-31-93-165 ~]$
```

59-Practice-2 - Platform independent script to find the java version

```
import subprocess
cmd=["bash","--version"]
sp=subprocess.Popen(cmd,shell=False,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_newlines=True)
rc=sp.wait()
o,e=sp.communicate()

if rc==0:
   for each_line in o.splitlines():
      if "version" in each_line  and "release" in each_line:
         print(each_line.split()[3].split('(')
else:
   print("Command was failed and error is: ",e)
```

Before code output:

```
[ec2-user@ip-172-31-93-165 ~]$ bash --version
GNU bash, version 4.2.46(2)-release (x86_64-redhat-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[ec2-user@ip-172-31-93-165 ~]$
```

After code output:

```
[ec2-user@ip-172-31-93-165 ~]$ python3 bash_version.py
4.2.46
```

Try for java –version

```
[ec2-user@ip-172-31-93-165 ~]$ java -version
java version "1.7.0_211"
OpenJDK Runtime Environment (amzn-2.6.17.1.79.amzn1-x86_64 u211-b02)
OpenJDK 64-Bit Server VM (build 24.211-b02, mixed mode)
[ec2-user@ip-172-31-93-165 ~]$
```

14-Working with text files:

60-Working with text files - Reading and writing to text files

Creating an empty file

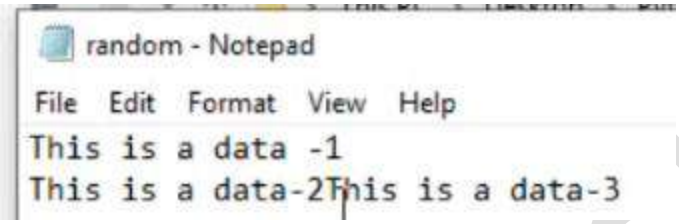
```
Create a new file
add content to an existing file
read a file content

open --> w  -> creation of an empty file
     --> a  -> add/append more content to an existing file
     --> r  -> just to read an existing file
#Creation of empty file
fo=open('newfile.txt','w')


fo.close() -> Close the file once your file
```

Create a new content to a file

```
fo=open("random.txt",'w')
fo.write("This is a new content\n")

fo.close()
```

Instead of individual write line we can write with a single line and provide the content to a writelines() function.

```
my_content=["This is a data -1\n","This is a data-2","This is a data-3"]
fo=open("random.txt",'w')
'''
fo.write("This is a first line\n")
fo.write("This is a second line\n")
fo.write("This is a third line")
'''
fo.writelines(my_content)
fo.close()
```



Writing files with loops

```
my_content=['This is using loop-iteratioin-1','This is using loop-iterantion-2','This is using loop-i

fo=open("with_loop.txt",'w')

for each_line in my_content:
    fo.write(each_line+"\n")
fo.close()
```

To append the data to the same file

```
my_content=['This is using loop-iteratioin-1','This is using loop-iterantion-2','This is using loop-i

fo=open("with_loop.txt",'a')

for each_line in my_content:
    fo.write(each_line+"\n")
fo.close()
```

To read the data of the same file

```
fo=open("with_loop.txt","r")
print(fo.read())
fo.close()
```

```
fo=open("with_loop.txt","r")
print(fo.readline())
fo.close()
```

To read the data of all lines at a time but need to print line by line

```
fo=open("with_loop.txt","r")
data=fo.readlines()
fo.close()

print(data)
```

```
['This is using loop-iteratioin-1\n', 'This is using loop-iterantion-2\n', 'This is using loop-iteratioin-3\n', 'This is using
loop-iteratioin-1\n', 'This is using loop-iterantion-2\n', 'This is using loop-iteratioin-3\n']
[Finished in 0.8s]
```

```
fo=open("with_loop.txt","r")
data=fo.readlines()
fo.close()

for each in range(3):
    print(data[each])    #data[0], data[1],data[2]
```

```
This is using loop-iteratioin-1

This is using loop-iterantion-2

This is using loop-iteratioin-3

[Finished in 1.1s]
```

To get the last value of your data

```
fo=open("with_loop.txt","r")
data=fo.readlines()
fo.close()
'''
for each in range(3):
    print(data[each])    #data[0], data[1],data[2]
'''
print(data[-1])
```

61-Copy the content of a source file into a destination file

```
sfile="random.txt"
dfile="newrandom.txt"

sfo=open(sfile,'r')
content=sfo.read()
sfo.close()

dfo=open(dfile,'w')
dfo.write(content)
dfo.close()
```

Providing paths to source file and destination files variables.

```
sfile="C:\\Users\\Automation\\Desktop\\random.txt"
dfile="C:\\Users\\Automation\\Downloads\\newrandom.txt"

sfo=open(sfile,'r')
content=sfo.read()
sfo.close()

dfo=open(dfile,'w')
dfo.write(content)
dfo.close()
```

User inputs to source and destination variables

```
sfile=input("Enter your source file: ")
dfile=input("Enter your destination file: ")
sfo=open(sfile,'r')
content=sfo.read()
sfo.close()

dfo=open(dfile,'w')
dfo.write(content)
dfo.close()
```

By default it is a read mode

```
sfo=open(sfile)
print(sfo.mode)
```

15-Working with csv (comma separated values):

To store tabular type of data

Import csv

read_csv_file.py

```
import csv
req_file="C:\\Users\\Automation\\Desktop\\hi\\info.csv"

fo=open(req_file,"r")
content=fo.readlines()
fo.close()

for each in content:
    print(each.strip("\n"))
```

```
S_NO,Name,Salary,Skill
1,John,2000$,Python
2,Json,1000$,Bash
[Finished in 0.9s]
```

As per our output we are getting output as a string

Now we have to convert to tabular form, that is in rows and columns

```
fo=open(req_file,"r")
data=csv.reader(fo)

fo.close()
```

We will use csv.reader()

```
import csv
req_file="C:\\Users\\Automation\\Desktop\\hi\\info.csv"

fo=open(req_file,"r")
data=csv.reader(fo)

for each in data:
    print(each)
fo.close()
```

```
['S_NO', 'Name', 'Salary', 'Skill']
['1', 'John', '2000$', 'Python']
['2', 'Json', '1000$', 'Bash']
[Finished in 1.0s]
```

```
import csv
req_file="C:\\Users\\Automation\\Desktop\\hi\\info.csv"

fo=open(req_file,"r")
content=fo.readlines()
fo.close()

for each in content:
    print(each.strip('\n').split(","))
```

```
['S_NO', 'Name', 'Salary', 'Skill']
['1', 'John', '2000$', 'Python']
['2', 'Json', '1000$', 'Bash']
[Finished in 1.0s]
```

62-Introduction to csv files and How to read a csv files using python

To give the delimiter for csv to recognize with our custom delimiter

```
import csv
req_file="C:\\Users\\Automation\\Desktop\\hi\\info.csv"

fo=open(req_file,"r")
content=csv.reader(fo,delimiter=",")
for each in content:
    print(each)

fo.close()
```

```
['S_NO', 'Name', 'Salary', 'Skill']
['1', 'John', '2000$', 'Python']
['2', 'Json', '1000$', 'Bash']
[Finished in 1.0s]
```

63-Read only header of a csv file and Finding the no of rows in a csv file

```
read_csv_file.py
1  import csv
2  req_file="C:\\Users\\Automation\\Desktop\\hi\\new_ir
3
4  fo=open(req_file,"r")
5  content=csv.reader(fo,delimiter="|")
6  #print(list(content))
7  #print(f'The header is:\n {list(content)[0]}')
8  header=next(content)
9  print("The header is: ",header)
10 print(list(content))
11 '''
12 for each in content:
13     print(each)
14 '''
15 fo.close()
```

```
The header is:  ['S_NO', 'Name', 'Salary', 'Skill']
[['1', 'John', '2000$', 'Python'], ['2', 'Json', '1000$', 'bash']]
[Finished in 0.8s]
```

```
print("The no of rows are: ",len(list(content)))
'''
```

```
The header is:  ['S_NO', 'Name', 'Salary', 'Skill']
The no of rows are:  2
[Finished in 0.7s]
```

64-Creating csv file

```
req_file="C:\\Users\\Automation\\Desktop\\hi\\demo.csv"
fo=open(req_file,'w')
csv_writer=csv.writer(fo,delimiter=",")
csv_writer.writerow(['S_No',"Name",'Age'])

fo.close()
```

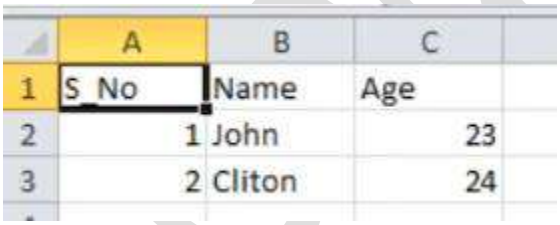
```
    1    2    3
1 S_No,Name,Age
2
```

Writing a newline to csv file

```
req_file="C:\\Users\\Automation\\Desktop\\hi\\demo.csv"
fo=open(req_file,'w',newline="")
csv_writer=csv.writer(fo,delimiter=",")
csv_writer.writerow(['S_No',"Name",'Age'])
csv_writer.writerow([1,"John",23])
csv_writer.writerow([2,"Cliton",24])
fo.close()
```

	A	B	C	D
1	S_No	Name	Age	
2	1	John	23	
3	2	Cliton	24	
4				

```
req_file="C:\\Users\\Automation\\Desktop\\hi\\demo.csv"
fo=open(req_file,'w',newline="")
csv_writer=csv.writer(fo,delimiter=",")
'''
csv_writer.writerow(['S_No',"Name",'Age'])
csv_writer.writerow([1,"John",23])
csv_writer.writerow([2,"Cliton",24])
'''
my_data=[['S_No',"Name",'Age'],[1,"John",23],[2,"Cliton",24]]
csv_writer.writerows(my_data)
fo.close()
```



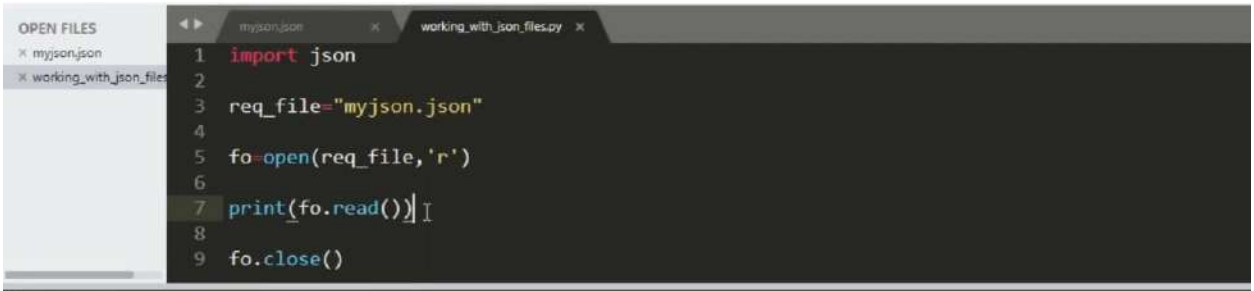
	A	B	C
1	S_No	Name	Age
2	1	John	23
3	2	Cliton	24

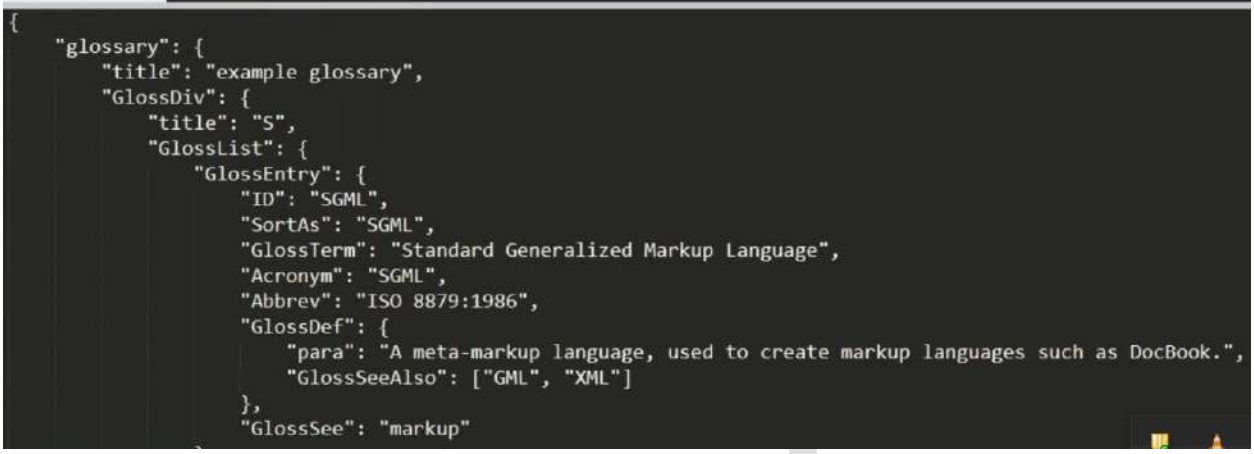
16-Working with files like json:

- To transfer data from server to server or server to client and vice-versa
- For simple things we are using json to transfer data
- For complex things we are using XML to transfer data

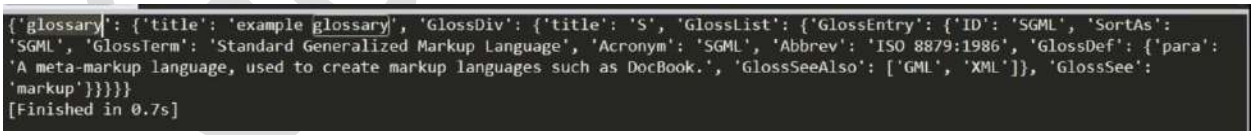
65-Working with json files

Reading a json data and process as a dictionary





Convert above data into python data.



Write your data to a json file

```
my_dict={'Name':'Narendra','skills':['Python','shell','yaml','AWS']}

req_file="myinfo.json"

fo=open(req_file,'w')
json.dump(my_dict,fo)

fo.close()
```

To get the data and indent of your generating json file

```
json.dump(my_dict,fo,indent=4)
```

```
my_dict={'Name':'Narendra','skills':['Python','shell','yaml','AWS']}

req_file="myinfo.json"

fo=open(req_file,'w')
json.dump(my_dict,fo,indent=4)

fo.close()
```

Cat myinfo.json

To get data from a dictionary variable

print(win_data)
print(win_data['current_login_user'][0])
print(win_data['Host_info']['IP'])

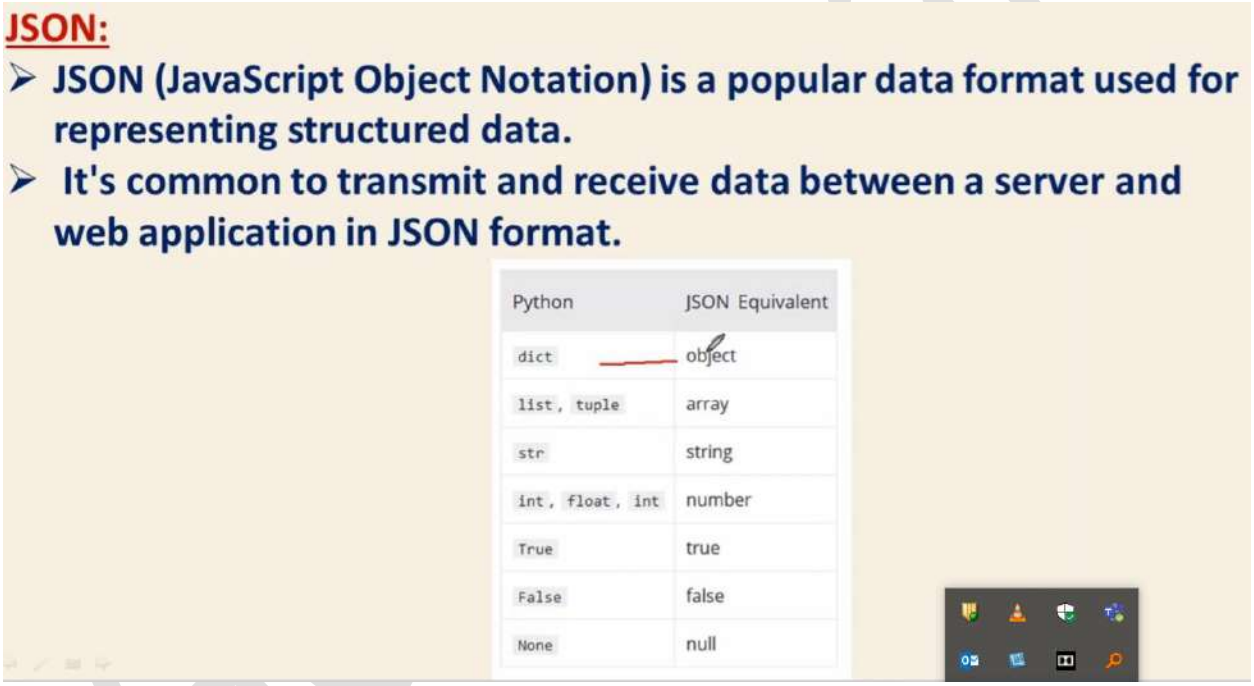
Loop along dictionary keys

```
for key in win_data:
   print(f'{key} : {win_data[key]}')
```

load json file

```
with open("logs\\host_output.json") as a:
   dict1 = json.load(a)
print(dict1["Host_info"]["FQDN"])
```

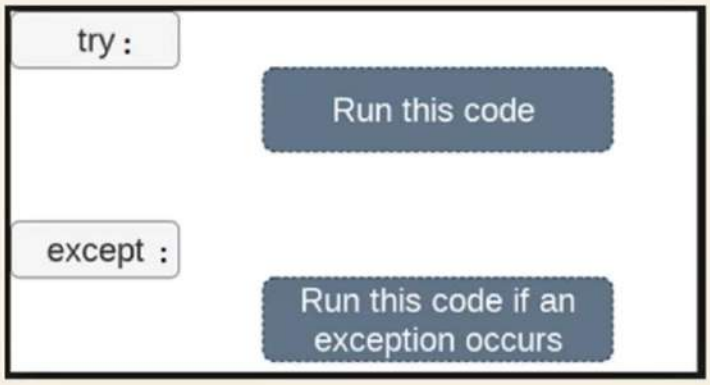
update json file once loaded the json file



17-Exception Handling:

How to handle Exceptions?

- **Exceptions can be handled using try and except statement.**
- **Syntax:**

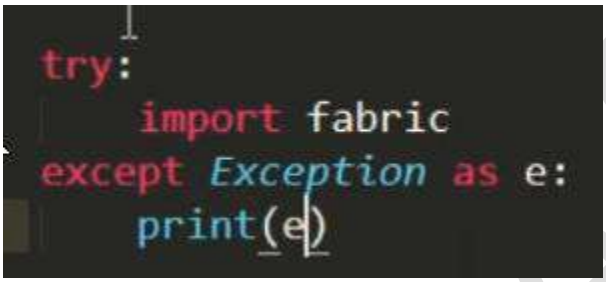


66-Introduction to Exception Handling

```
print("Welcome to exceptions concept")
print("Now it is fine")

try:
    fo=open("nari.txt")
    print(fo.read())
    fo.close()
except Exception as e:
    print(e)
```

```
Welcome to exceptions concept
Now it is fine
[Errno 2] No such file or directory: 'nari.txt'
[Finished in 0.9s]
```

```
Runtime Errors--> Exceptions, There is a way to handle runtime errors.
How to handle Exceptions?
Ans:
    with try eand except block

    try:
        statement-1
        statement-2
        statement-3
    except:
        print("This is because of of something went wrong in try block")
Useful:

    try:
        statement-1
        statement-2
        statement-3
    except Exception as e:
        print("This is beacuse of ",e)
```



Exceptions Examples

67-Exception handling for known Exceptions

```
#NameError
#TypeError
#FileNotFoundError
#ZeroDivisionError

try:
    #print(a)
    print(4+"hi")
except NameError:
    print("Variable is not defined")
except Exception as e:
    print(e)
```

If any one of the line error out, then all remaining lines will not be executed.

```
try:
    print(a)
    #print(4+"hi")
    #open('asdfas.txt')
    #print(5/0)
    import fabric
except FileNotFoundError:
    print("File is not present to open it")
except NameError:
    print("Variable is not defined")
except TypeError:
    print("Adding number and string is not possible")
except ZeroDivisionError:
    print("Division with zero is not possible")
except ModuleNotFoundError:
```

```
try:
    print("thsi is simple exceptions")
    print(a)
except Exception as e:
    print(e)
 ========================================
try:
    print("thsi is simple exceptions")
    print(a)
except NameError:
    print("variable is not define")
except Exception as e:
    print(e)
finally:
    print("This will execute after trying with try and except blocks")
```

68-try except else and finally usage

```
try:
    print("thsi is simple exceptions")
    print(a)
except Exception as e:
    print(e)
 ========================================
try:
    print("thsi is simple exceptions")
    print(a)
except NameError:
    print("variable is not define")
except Exception as e:
    print(e)
finally:
    print("This will execute after trying with try and except blocks")
```

```
try:
    a=10
    print(a)
except NameError:
    print("Variable is not defined")
except Exception as e:
    print("Exception occured:",e)
finally:
    print("This will executes always")
```

```
try:
    print(a)
except NameError:
    print("Variable is not defined")
except Exception as e:
    print("Exception occured:",e)
else:
    print("This will execute if there is no exceptions in try block")
```

69-Raise user Defined Exceptions

We can create custom Exceptions using:

raise (Used to raise an existing exceptions)

assert (Used to create an AssertionError)

Raise <exceptionError>

```
age=23

if age>30:
    print("Valid age")
else:
    raise ValueError("Age is less than 30")
```

Assert error:

AssertionError, you have to assume that, wantedly created exception.

```
age=33

try:
    assert age<30
    print("Valid age")
except:
    print("Exception occured")
```

```
age=20

try:
    assert age>30
    print("Valid age")
except AssertionError:
    print("Raised with assert because age is lessthan 30")
except:
    print("Exception occured")
```

```
Raised with assert because age is lessthan 30
[Finished in 0.5s]
```

18-Functions:

70-Introduction to Functions

You can use functions, unless you have same logic in different places.
Whatever your logic is repeating, just create one function and call it wherever you want.

Function:

- **A function is a block of code for some specific operation.**
- **Function code is re-usable.**
- **A function executes only when it is called.**

```
import os


print("Please wait. Cleaning the screen....")
os.system("cls")
print("Please wait finding the list of dir and files")
os.system("dir")
```

run_your_commands.py

```
import os
import time
import platform

if platform.system()=="Windows":
    print("Please wait. Cleaning the screen....")
    time.sleep(2)
    os.system("cls")
    print("Please wait finding the list of dir and files")
    time.sleep(2)
    os.system("dir")
else:
    print("Please wait. Cleaning the screen....")
    time.sleep(2)
    os.system("clear")
    print("Please wait finding the list of dir and files")
    time.sleep(2)
    os.system("ls -lrt")
```

run_your_commands.py

```
import os
import time
import platform
def mycode(cmd1,cmd2):
    print("Please wait. Cleaning the screen....")
    time.sleep(2)
    os.system(cmd1)
    print("Please wait finding the list of dir and files")
    time.sleep(2)
    os.system(cmd2)

if platform.system()=="Windows":
    mycode("cls","dir")
else:
    mycode('clear','ls -lrt')
```

71-How to define a Function and How to use defined Function, Types of Functions

```
#Rules to define function name:
#Function name should have only a-z,A-Z,0-9, _
#Function shuold not start with number but can it be start with _
#Dont include any space.
#Function must be define befor calling it
```

72-Converting simple code into Functions

demo.py | inter_fun.py

```
def welcome_msg():
    print("Welcome to Python Scripting")
    print("Python is easy to learn")
    return None
def known_concepts():
    print("Now we are good with bascis")
    print("We are about to start functions concepts in python")
    return None
def new_concepts():
    print("Function are very easy in python")
    print("Now we are writing simple functions")
    return None
welcome_msg()
known_concepts()
new_concepts()
```

73-Calling a function from another function and Scope of the variables

Scope of the variable:

- ✓ **The location where we can find a variable and also access it if required is called the scope of a variable.**
- ✓ **Local and Global variables.**

Defining a global variables for calling globally.

```
def myfunction1():
    x=60  #This is local variable
    print("Welcome to functions")
    print("x value from fun1: ",x)
    myfunction2()
    return None


def myfunction2():
    print("Thank you!!")
    print("x value from fun2: ",x)
    return None


x=10 #This is a global variable
myfunction1()

Welcome to functions
x value from fun1:  60
Thank you!!
x value from fun2:  10
[Finished in 0.1s]
```

Passing values from main() to other functions()

```
def myfunction2():
    print("Thank you!!")
    print("x value from fun2: ",x)
    return None

def main():
    global x
    x=10
    myfunction1()
    return None


main()

Welcome to functions
x value from fun1:  60
Thank you!!
x value from fun2:  10
[Finished in 0.1s]
```

making your variable global is not always good, sometime you can take it.

we have other way to pass a variable

```
def myfunction2(x):
    print("Thank you!!")
    print("x value from fun2: ",x)
    return None

def main():
    #global x
    x=10
    myfunction1()
    myfunction2(x)
    return None


main()
```

```
 9  def myfunction2(y):
10      print("Thank you!!")
11      print("x value from fun2: ",y)
12      return None
13
14  def main():
15      #global x
16      x=10
17      myfunction1()
18      myfunction2(x)
19      return None
20
21
22  main()
```

```
Welcome to functions
x value from fun1:  60
Thank you!!
x value from fun2:  10
[Finished in 0.1s]
```

```
 9  def myfunction2(y): #Parameter
10      print("Thank you!!")
11      print("x value from fun2: ",y)
12      return None
13
14  def main():
15      #global x
16      x=10 variable
17      myfunction1()
18      myfunction2(x) #Argument
19      return None
20
21
22  main()
```

74-Simple Functions with arguments

Differences between variable, argument & parameter

```
def myfunction2(y):  #Parameter or positional argument
    print("Thank you!!")
    print("x value from fun2: ",y)
    return None

def main():
    #global x
    x=10  variable
    myfunction1()
    myfunction2(x)  #Argument
    return None


main()
```

```
def get_result(value):
    result=value+10
    print(f'Your result is: {result}')
    return None
def main():
    #global num
    num=eval(input("Enter your number: "))
    get_result(num)  #
    return None

main()
```

```python
def get_add(a,b):
    aresult=a+b
    print(f'The addition of {a} and {b} is: {aresult}')
    return None
def get_sub(a,b):
    sresult=a-b
    print(f'The sub of {a} and {b} is: {sresult}')
    return None

def main():
    a=eval(input("Enter your f
    b=eval(input("Enter your s
    get_add(a,b)
    get_sub(b,a)
    return None

main()
```

```
C:\Users\Automation\Desktop\Python Practice>python function_with_simple_arguments.py
Enter your first num: 4
Enter your second num: 7
The addition of 4 and 7 is: 11
The sub of 4 and 7 is: -3

C:\Users\Automation\Desktop\Python Practice>python function_with_simple_arguments.py
Enter your first num: 4
Enter your second num: 7
The addition of 4 and 7 is: 11
The sub of 7 and 4 is: 3

C:\Users\Automation\Desktop\Python Practice>
```

Before a,b position

After b,a position

75-Functions with arguments and return value

```python
def get_addition(a,b):
    result=a+b
    #print(f"The addition of {a} and {b} is: {result}")
    return result
def main():
    a=eval(input("Enter your first number: "))
    b=eval(input("Enter your second number :"))
    result=get_addition(a,b)
    print(f"The addition of {a} and {b} is: {result}")
    return None
main()
```

```
C:\Users\Automation\Desktop\Python Practice>python fun_with_arg_return_values.py
Enter your first number: 3
Enter your second number :4
The addition of 3 and 4 is: None

C:\Users\Automation\Desktop\Python Practice>python fun_with_arg_return_values.py
Enter your first number: 4
Enter your second number :5
The addition of 4 and 5 is: 9
```

```
def multiply_num_10(value):
    #result=value*10
    #return result
    return value*10


def main():
    num=eval(input("Enter your number: "))
    result=multiply_num_10(num)
    print("The result is: ",result)



main()
```

76-Functions with default arguments

```
1  def display(a=1):
2      print("The value of a is: ",a)
3      return None
4
5  display(4)
6  display(5)
7  display()
```

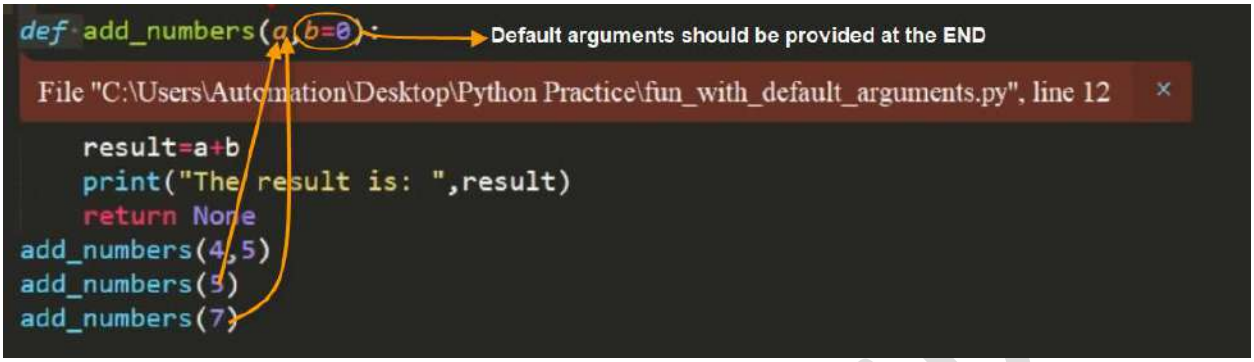
default value to your argument

```
The value of a is:  4
The value of a is:  5
The value of a is:  1
[Finished in 0.2s]
```

```
12 def add_numbers(a,b=0):
13     result=a+b
14     print("The result is: ",result)
15     return None
16 add_numbers(4,5)
17 add_numbers(5)
18
```

```
The result is:  9
The result is:  5
[Finished in 0.2s]
```

Default argument should be provided at the END



Error of non-default argument

```
SyntaxError: non-default argument follows default argument
[Finished in 0.2s with exit code 1]
```

Note:

If you don't pass the "user", it will take the **default** "user",
If you pass, it will take the passed "user"

```
def working_on_some(user="root"):
    print(f"working with {user}")
    return None

working_on_some("weblogic_admin")
```

```
working with weblogic_admin
[Finished in 0.1s]
```

77-Functions with keyword-based arguments

```
def display(a,b):
    print(f'a={a}')
    return None

display(3,4)
display(a=3,b=4)
display(b=4,a=3)
```

```
a=3
a=3
a=3
[Finished in 0.1s]
```

78-Functions with Variable length arguments (*arg)

```
def display(a):
    print(type(a))
    return None
display(4)
display(4,5)
```

```
File "C:\Users\Automation\Desktop\Python Practice\functions_with_variable_length_argument.py", line 5, in <module>
```

```
<class 'int'>Traceback (most recent call last):

  File "C:\Users\Automation\Desktop\Python Practice\functions_with_variable_length_argument.py", line 5, in <module>
    display(4,5)
TypeError: display() takes 1 positional argument but 2 were given
[Finished in 0.2s with exit code 1]
```

So we have a syntax to pass our argument to accept any length of our passed arguments to our functions

```
def display(*data):
    print(data)
    return None
display()
display(4)
display(4,5,67)
```

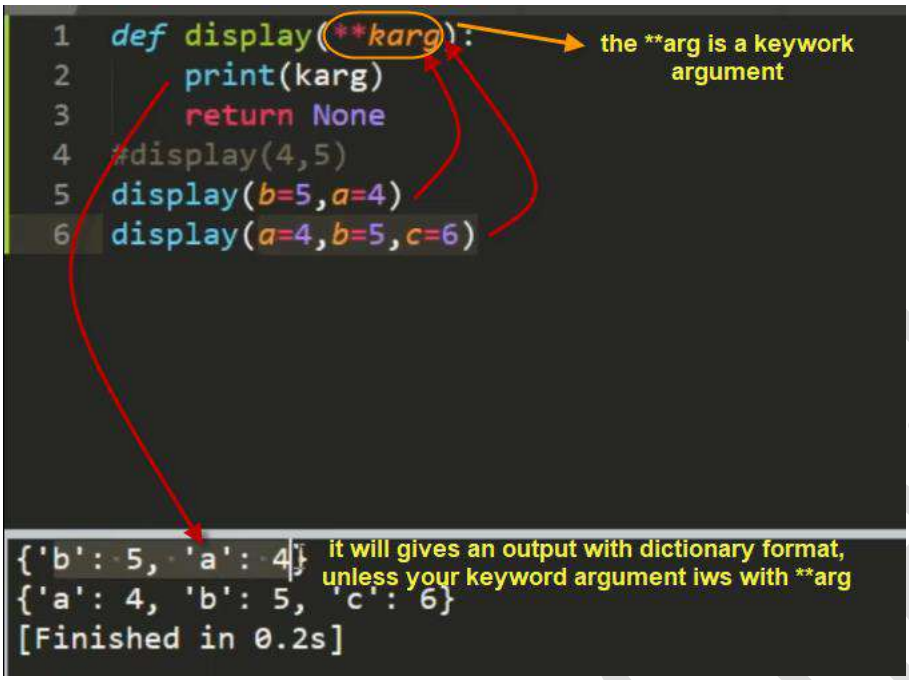
```
()
(4,)
(4, 5, 67)
[Finished in 0.2s]
```

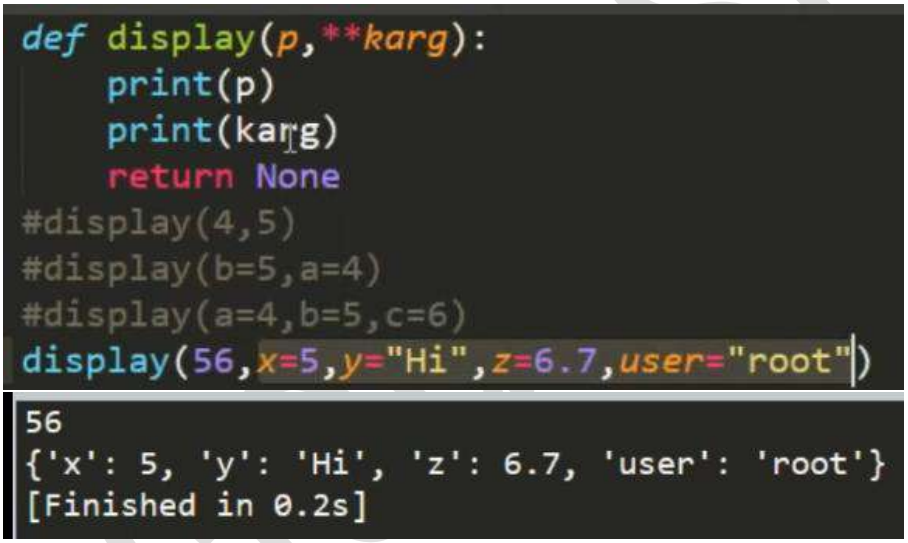
```
def display(*arg):
    print(arg)
    return None
display()
display(4)
display(4,5,67)
```

```
def display(*arg):
    for each in arg:
        print(type(each))
    return None
#display()
#display(4)
display(4,5,67)
print('-------------')
display("hi",4.65)
```

```
<class 'int'>
<class 'int'>
<class 'int'>
-------------
<class 'str'>
<class 'float'>
[Finished in 0.2s]
```

79-Functions with variable keyword arguments (**arg)





80-How to use Functions of one script into another script, what is _name_

Script1.py

```
def addition(a,b):
    print(f"The addition of {a} and {b} is {a+b}")
    return None
def sub(a,b):
    print(f'The sub of {a} and {b} is {a-b}')
    return None


x=7
y=4
addition(x,y)
sub(x,y)
```

Script2.py

```
def mult(a,b):
    print(f'The mul of {a} and {b} is {a*b}')
    return None

x=10
y=20
mult(x,y)
```

From Script1.py in need to call values inside script2.py

script1.py

```
def addition(a,b):
    print(f"The addition of {a} and {b} is {a+b}")
    return None
def sub(a,b):
    print(f'The sub of {a} and {b} is {a-b}')
    return None

def main():
    x=7
    y=4
    addition(x,y)
    sub(x,y)
    return None
if __name__=="__main__":
    main()
```

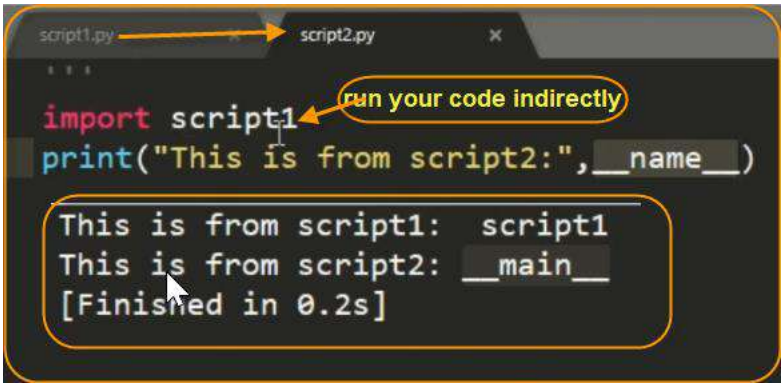
script2.py

```
import script1

def mult(a,b):
    print(f'The mul of {a} and {b} is {a*b}')
    return None
def main():
    x=10
    y=20
    script1.addition(x,y)
    #script1.sub(x,y)
    #mult(x,y)
    return None
if __name__=="__main__":
    main()
```

```
import script1

print(dir(script1))
```

```
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__',
'addition', 'sub']
The mul of 10 and 20 is 200
[Finished in 0.1s]
```

Running your code indirectly with the help of import command.



81-Simple exception handling to changing current working directory

At first list out the possibility exceptions

```
NotADirectoryError: [Errno 20] Not a directory: '/tmp/hi.txt'
FileNotFoundError: [Errno 2] No such file or directory: '/tmp/hello.txt'
PermissionError: [Errno 13] Permission denied: '/etc/httpd/conf'
```

```
import os
req_path=input("Enter path to change working dir: ")
print("The current working dir is: ",os.getcwd())
try:
  os.chdir(req_path)
  print("Now your new working dir is: ",os.getcwd())
except FileNotFoundError:
  print("Given path is not a valid path. So cant change working directory")
except NotADirectoryError:
  print("Given path is a file path. So cant change working directory")
except PermissionError:
  print("Sorry you dont have access for the given path. So cant chagne working directory")
except Exception as e:
  print(e)
```

Writing above code in function

```
def main():
  req_path=input("Enter path to change working dir: ")
  print("The current working dir is: ",os.getcwd())
  try:
    os.chdir(req_path)
    print("Now your new working dir is: ",os.getcwd())
  except FileNotFoundError:
    print("Given path is not a valid path. So cant change working directory")
  except NotADirectoryError:
    print("Given path is a file path. So cant change working directory")
  except PermissionError:
    print("Sorry you dont have access for the given path. So cant chagne working directory")
  except Exception as e:
    print(e)
  return None


if __name__=="__main__":
   main()
-- INSERT --
```

19-Regular expression with re module:

82-Introduction to regular expressions

What is a Regular Expression (RegEx) ?

- The regex is a procedure in any language to look for a specified pattern in a given text.
- What is a pattern?
  - It is a sequence of characters, which represent multiple strings
  - Ex: 'i[st]' --> is, it
    - 'python[23]' --> python2, python3

```
>>> my_str="Python is simple and it easy"
>>> my_str.split("is")
['Pyth  ', ' simple and it easy']
>>> my_str.split("it")
['Python is simple and ', ' easy']
>>> import re
>>> re.split("i[st]",my_str)
['Python ', ' simple and ', ' easy']
>>>
```

Python module for Regular Expressions : re

- The different operations in python re are:
  - match()
  - search()
  - findall()
  - finditer()
  - sub()
  - split()
  - compile() etc...

83-Basic rules to create a pattern for regex

Regular Expressions (RegEx):

- The regex is a procedure in any language to look for a specified pattern in a given text.

- re is the module to perform regex in Python. (use import re in scripts)
- There are different operations in re like:
  - search, match, finditer, findall, split, sub etc...

- Syntax:
  - re.search(pattern, text)
  - re.match(pattern, text)
  - re.finditer(pattern, text)
  - re.findall(pattern, text)
- Pattern is a sequence of characters, which represent multiple strings.
- Simple examples for pattern:
  - "Python"
  - "Python[23]" ➔ "Python2" "Python3"

Rules to create a pattern:

`a`, `X`, `9` - Ordinary characters that match themselves

`[abc]` - Matches `a` or `b` or `c`

`[a-c]` - Matches `a` or `b` or `c`

`[a-zA-Z0-9]` - Matches any letter from (`a` to `z`) or (`A` to `Z`) or (`0` to `9`)

`\w` - Matches any single letter, digit or underscore

`\W` - Matches any character not part of `\w`

`\d` - Matches decimal digit 0-9

`.` - Matches any single character except newline character

Note: Use \. to match .

For advance level for regular expression

- r"Python" or r"Python[23]"

Findall pattern

```
practice_for_regex.py
import re
text="This is a python and it is easy to learn"
my_pat="is"
print(re.findall(my_pat,text))
#print(re.findall("is","This is a python and it is easy to learn"))
```

```
['is', 'is', 'is']
['is', 'is', 'is']
[Finished in 0.1s]
```

To count the patterns of your result

```
print(len(re.findall(my_pat,text)))
```

```
my_pat="i[ts]"
print(re.findall(my_pat,text))
```

```
['is', 'is', 'it', 'is']
[Finished in 0.1s]
```

```
import re
text="This is a python and it @ is easy to learn @"
#my_pat="i[ts]"
'''
my_pat="[@as]"
print(re.findall(my_pat,text))
#print(len(re.findall(my_pat,text)))
#print(re.findall("is","This is a python and it is easy to learn"))
'''
my_pat="[a-d]" #"[abcd]"
print(re.findall(my_pat,text))
```

```
['a', 'a', 'd', 'a', 'a']
[Finished in 0.1s]
```

```
import re
text="This is a python and it @ is easy -to _ learn @"
```

```
my_pat="\w\w"
print(re.findall(my_pat,text))
```

```
['Th', 'is', 'is', 'py', 'th', 'on', 'an', 'it', 'is', 'ea', 'sy', 'to', 'le', 'ar', 'n_']
[Finished in 0.1s]
```

```
13  '''
14  my_pat="\w\w\w"
15  print(re.findall(my_pat,text))
16
['Thi', 'pyt', 'hon', 'and', 'eas', 'lea', 'rn_']
[Finished in 0.1s]
```

Capital W

```
18  my_pat="\W"
19  print(re.findall(my_pat,text))
[' ', ' ', ' ', ' ', ' ', ' ', '@', ' ', ' ', ' ', '-', ' ', ' ', '@']
[Finished in 0.1s]
```

Digit

```
import re
text="This is a python2 and 46 it @ is easy -to _ learn_@ python3"
```

```
my_pat="python2"
print(re.findall(my_pat,text))
```

Will be given as

```
my_pat="python\d"
print(re.findall(my_pat,text))
```

```
['python2', 'python3']
[Finished in 0.1s]
```

```
#my_pat="python\d"
my_pat="\d\d"
print(re.findall(my_pat,text))
```

```
['46']
[Finished in 0.1s]
```

.(dot) pattern

```
my_pat="."
print(re.findall(my_pat,text))
```

```
['T', 'h', 'i', 's', ' ', 'i', 's', ' ', 'a', ' ', 'p', 'y', 't', 'h', 'o', 'n', '2', ' ', 'a', 'n', 'd', ' ', '4',
'6', ' ', 'i', 't', ' ', '@', ' ', 'i', 's', ' ', 'e', 'a', 's', 'y', ' ', '-', 't', 'o', ' ', '_', ' ', 'l', 'e',
'a', 'r', 'n', '_', '@', ' ', 'p', 'y', 't', 'h', 'o', 'n', '3']
[Finished in 0.1s]
```

```
my_pat=".."
print(re.findall(my_pat,text))
```

```
['Th', 'is', ' i', 's ', 'a ', 'py', 'th', 'on', '2 ', 'an', 'd ', '46', ' i', 't ', '@ ', 'is', ' e', 'as', 'y ',
'-t', 'o ', '_ ', 'le', 'ar', 'n_', '@ ', 'py', 'th', 'on']
[Finished in 0.1s]
```

```
my_pat="..."
print(re.findall(my_pat,text))
```

```
['Thi', 's i', 's a', ' py', 'tho', 'n2 ', 'and', ' 46', ' it', ' @ ', 'is ', 'eas', 'y -', 'to ', '_ l', 'ear',
'n_@', ' py', 'tho']
[Finished in 0.1s]
```

```
28  my_pat="\."
29  print(re.findall(my_pat,text))
```

```
['.', '.']
[Finished in 0.1s]
```

```
text="This is my ip of a db server: 255.100.102.103"
pat="\d\d\d.\d\d\d.\d\d\d.\d\d\d"
print(re.findall(pat,text))
```

```
['255.100.102.103']
[Finished in 0.1s]
```

```
text="This is my ip of a db server: 255.100.102.103"
pat="\w\w\w.\w\w\w.\w\w\w.\w\w\w"
print(re.findall(pat,text))
```

```
['255.100.102.103']
[Finished in 0.1s]
```

```
27
28  #my_pat="\."
29  #print(re.findall(my_pat,text))
30
31  text="This is my ip of a db server: 255.100.102.103  2343534534534534535345"
32  pat="\d\d\d.\d\d\d.\d\d\d.\d\d\d"
33  print(re.findall(pat,text))
34
```

```
['255.100.102.103', '234353453453453']
[Finished in 0.1s]
```

```
text="This is my ip of a db server: 255.100.102.103  2343534534534534535345"
pat="\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d"
print(re.findall(pat,text))
```

```
['255.100.102.103']
[Finished in 0.1s]
```

84-Rules to create a pattern Part-2

Rules to create a pattern:

`a`, `X`, `9` - Ordinary characters that match themselves

`[abc]` - Matches `a` or `b` or `c`

`[a-c]` - Matches `a` or `b` or `c`

`[a-zA-Z0-9]` - Matches any letter from (`a` to `z`) or (`A` to `Z`) or (`0` to `9`)

`\w` - Matches any single letter, digit or underscore

`\W` - Matches any character not part of `\w`

`\d` - Matches decimal digit 0-9

`.` - Matches any single character except newline character

Note: Use `\.` to match .

Starting of the line with "^"

`^` - Start of the string (and start of the line in-case of multiline string)

```
import re
text='isa python and it is easy to learn'
my_pat='^i[ts]'
print(re.findall(my_pat,text))
```

```
['is']
[Finished in 0.1s]
```

End of the line with "$"

`$` - End of the string (and newline character in-case of multiline string)

```
import re
text='isa python andlearn it learn is easy to learn'
#my_pat='^i[ts]'
my_pat="learn$"
print(re.findall(my_pat,text))
```

```
['learn']
[Finished in 0.1s]
```

Empty string at the beginning "\b"

\b - Empty string at the beginning or end of a word

Starting of a word

```
my_pat="\blearn"
print(re.findall(my_pat,text))
```

```
[]
[Finished in 0.1s]
```

Ending of a word

```
import re
text='isa python and learn it learnis easy to learn'
#my_pat='^i[ts]'
#my_pat="learn$"
my_pat=r"\blearn\b"
print(re.findall(my_pat,text))
```

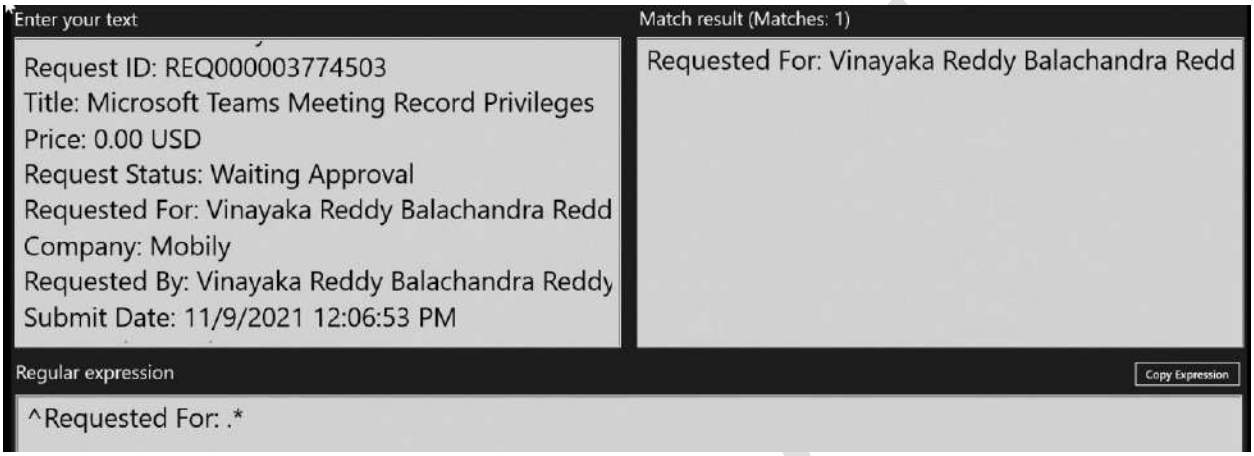
```
['learn', 'learn']
[Finished in 0.1s]
```

```
my_pat=r"learn"
print(re.findall(my_pat,text))
```

```
['learn']
[Finished in 0.1s]
```

About "?" and ".*"

With the non-greedy modifier (?), the .* expression will match as *soon* as either a hash sign or end-of-line is reached. The default is to match as *much* as possible, and that is why you always got the whole line.



Solution: \bBy.*

85-Rules to create a pattern Part-2

86-Regex with Flags

87-working with search and match operations from re module

88-working with findall and finditer operations

89-working with split, sub and subn operations of re module

90-compile operation (Execute all re operations on compile object)

20-Paramiko module to work with remote servers using python:

91-Introduction to paramiko

92-Transfer file from local server to remote server and vice versa using paramiko

21-Shutil module:

93-Part-1 - copy file(s) with shutil module

22-OOPS for Real Time:

100-Encapsulation

94-Introduction to oops

95-Class and object attributes

96-Constructor of a class

97-Simple Python Script without and with oops concepts

98-destructor of a class

99-Polymorphism and inheritance of python oops

23-Windows Subsystem for Linux:

101-Linux for Developers on Windows

24-Python Logging

Logging is similar in spirit to a Print Function. However, a Print function lacks a lot of information that might be useful to a developer. Logging, on the other hand, can log the timestamps and line number at which the error occurred. It can log errors or any information to files, sockets, etc., and it offers you five types of severity based on which you can differentiate your logging.

```
import logging

logging.basicConfig(format='Date-Time : %(asctime)s : Line No. : %(lineno)d - %(message)s', level =
logging.DEBUG)
```

File mode: logging.basicConfig(level = logging.INFO, filename = 'datacamp.log', filemode = 'w')

```
logging.debug("A Debug Logging Message")
logging.info("A Info Logging Message")
logging.warning("A Warning Logging Message")
logging.error("An Error Logging Message")
logging.critical("A Critical Logging Message")
```

Output:

```
Date-Time : 2021-11-08 11:21:02,715 : Line No. : 69 - extracted with the 'EXPIRED' value
Date-Time : 2021-11-08 11:21:02,715 : Line No. : 69 - extracted with the 'EXPIRED' value
Date-Time : 2021-11-08 11:21:02,715 : Line No. : 69 - extracted with the 'EXPIRED' value
Date-Time : 2021-11-08 11:21:02,772 : Line No. : 90 - sending email....with to: 'g.shaik@mobily.com.sa' cc:
Date-Time : 2021-11-08 11:21:02,987 : Line No. : 92 - email sent...!
:: 2021-11-08 11:23:56,003 : :: : 33 - C:\Py_Projects\QA_DevOps_Tools_Licenses\Input\QA_DevOps_Tools_License
:: 2021-11-08 11:23:56,004 : :: : 44 - configfile.ini has been reaed to config obj
```

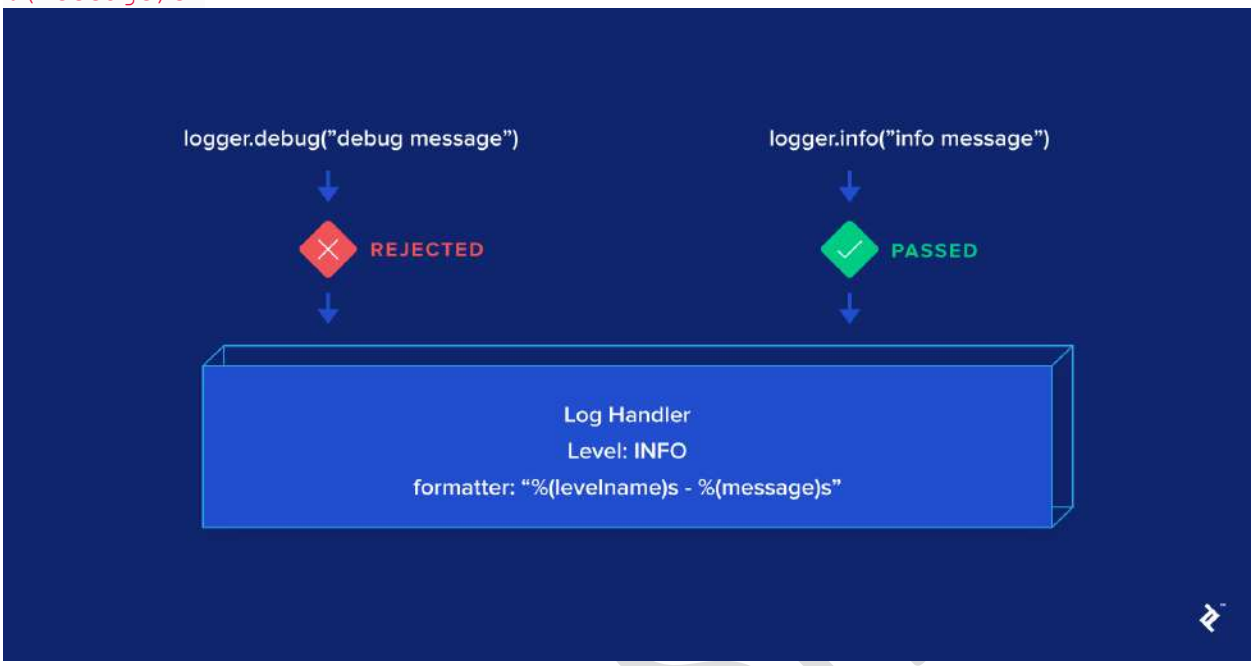
Five Levels of Logging as shown

- **Debug (10)**: Useful for diagnosing issues in the code.
- **Info (20)**: It can act as an acknowledgment that there are no bugs in the code. One good use-case of Info level logging is the progress of training a machine learning model.
- **Warning (30)**: Indicative of a problem that could occur in the future. For example, a warning of a module that might be discontinued in the future or low-ram warning.
- **Error (40)**: A serious bug in the code, could be a syntax error, out of memory error, exceptions.
- **Critical (50)**: An error due to which the program might stop functioning or might exit abruptly.

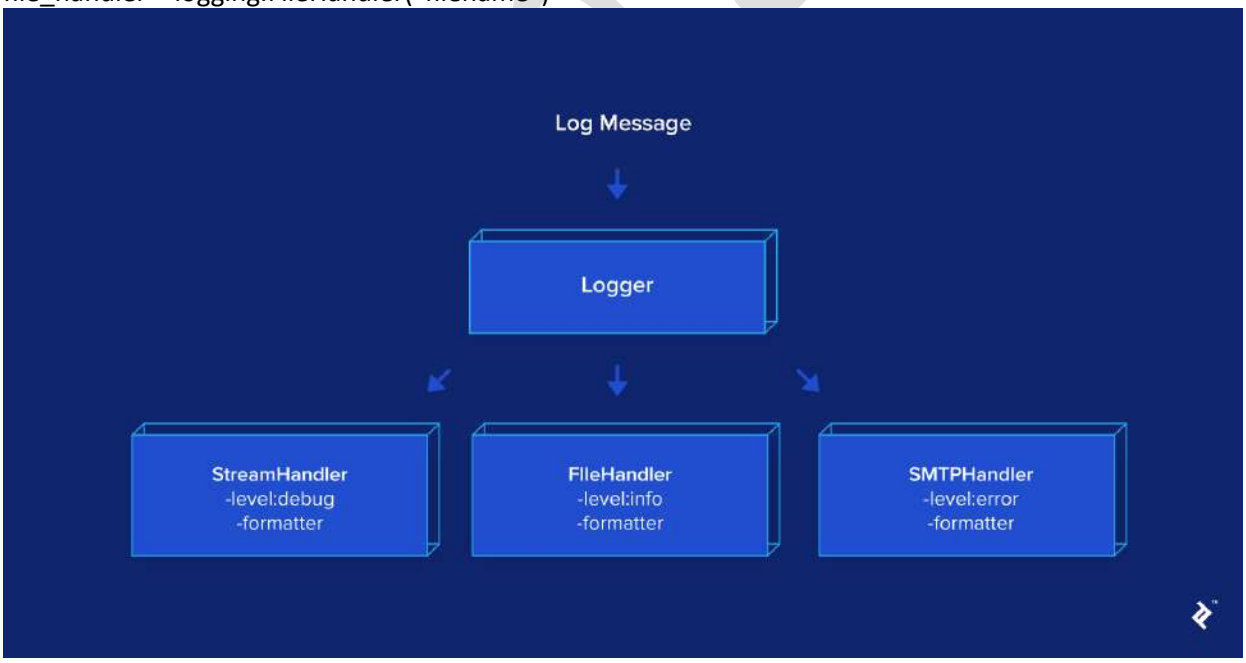
Attributes and formats

Attribute name	Format	Description
args	You shouldn't need to format this yourself.	The tuple of arguments merged into `msg` to produce `message`, or a dict whose values are used for the merge (when there is only one argument, and it is a dictionary).
asctime	`%(asctime)s`	Human-readable time when the `LogRecord` was created. By default this is of the form '2003-07-08 16:49:45,896' (the numbers after the comma are millisecond portion of the time).
created	`%(created)f`	Time when the `LogRecord` was created (as returned by `time.time()`).
exc_info	You shouldn't need to format this yourself.	Exception tuple (à la `sys.exc_info`) or, if no exception has occurred, `None`.
filename	`%(filename)s`	Filename portion of `pathname`.
funcName	`%(funcName)s`	Name of function containing the logging call.
levelname	`%(levelname)s`	Text logging level for the message (`'DEBUG'`, `'INFO'`, `'WARNING'`, `'ERROR'`, `'CRITICAL'`).
levelno	`%(levelno)s`	Numeric logging level for the message (`DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`).
lineno	`%(lineno)d`	Source line number where the logging call was issued (if available).
message	`%(message)s`	The logged message, computed as `msg % args`. This is set when `Formatter.format()` is invoked.
module	`%(module)s`	Module (name portion of `filename`).
msecs	`%(msecs)d`	Millisecond portion of the time when the `LogRecord` was created.
msg	You shouldn't need to format this yourself.	The format string passed in the original logging call. Merged with `args` to produce `message`, or an arbitrary object (see Using arbitrary objects as messages).
name	`%(name)s`	Name of the logger used to log the call.
pathname	`%(pathname)s`	Full pathname of the source file where the logging call was issued (if available).
process	`%(process)d`	Process ID (if available).
processName	`%(processName)s`	Process name (if available).
relativeCreated	`%(relativeCreated)d`	Time in milliseconds when the LogRecord was created, relative to the time the logging module was loaded.
stack_info	You shouldn't need to format this yourself.	Stack frame information (where available) from the bottom of the stack in the current thread, up to and including the stack frame of the logging call which resulted in the creation of this record.
thread	`%(thread)d`	Thread ID (if available).
threadName	`%(threadName)s`	Thread name (if available).

```
"%(asctime)s — %(name)s — %(levelname)s — %(funcName)s:%(lineno)d —
%(message)s"
```



The most common ones are StreamHandler and FileHandler:
console_handler = logging.StreamHandler()
file_handler = logging.FileHandler("filename")



Offline python modules download and install on remote server with pip:

Please refer: https://pip.pypa.io/en/stable/cli/pip_download/

Download module/package

Syntax: pip download <moduleName> -d"full/dir/path"

pip download schedule -d"C:\Users\gowse\OneDrive\Desktop\repo"

Copy to source to remote[offline]with the form of tar or local repo

copy downloaded module to offile remote server in one modules directory

Install module/package

Syntax: pip install <pythonmodulefiles> -f ./ --no-index --no-deps

pip install schedule-1.1.0-py2.py3-none-any.whl -f ./ --no-index --no-deps

1) Create a requirements.txt file with similar content (Note - these are the libraries you wish to download):

```
Flask==0.12
requests>=2.7.0
scikit-learn==0.19.1
numpy==1.14.3
pandas==0.22.0
```

One option for creating the requirements file is to use pip freeze > requirements.txt. This will list all libraries in your environment. Then you can go in to requirements.txt and remove un-needed ones.

2) Execute command mkdir wheelhouse && pip download -r requirements.txt -d wheelhouse to download libs and their dependencies to directory wheelhouse

3) Copy requirements.txt into wheelhouse directory

4) Archive wheelhouse into wheelhouse.tar.gz with tar -zcf wheelhouse.tar.gz wheelhouse

Then upload wheelhouse.tar.gz to your target machine:

1) Execute tar -zxf wheelhouse.tar.gz to extract the files

2) Execute pip install -r wheelhouse/requirements.txt --no-index --find-links wheelhouse to install the libs and their dependencies

On remote

For creation of virtual env testproject

```
python -m venv c:\path\to\myenv
cd c:\path\to\myenv\Scripts
activated.bat
```

pip install -r wheelhouse/requirements.txt --no-index --find-links wheelhouse

Create a requirement.txt file

pip freeze > requirements.txt

On the system that has access to internet

The pip download command lets you download packages without installing them:

pip download -r requirements.txt

(In previous versions of pip, this was spelled pip install --download -r requirements.txt.)

On the system that has no access to internet

Then you can use

pip install --no-index --find-links /path/to/download/dir/ -r requirements.txt

to install those downloaded modules, without accessing the network.

Another option is to use "pip wheel" to download and build all your dependencies, then you drop the pile of wheels on the target system and install into a virtualenv there. It doesn't gain you much over a tarball of the env except that it's upgradeable (which may not matter) and it's portable (virtualenvs only work in the directory they were originally created for, you can't move them easily).

./venv1/bin/pip wheel -w build/ [-r requirements.txt or ./yourapp]

virtualenv venv2

./venv2/bin/pip install --no-index -f ./wheels [-r requirements.txt or ./yourapp]

Some packaging tools for python

https://python-poetry.org/
https://pex.readthedocs.io/en/latest/
https://github.com/nylas/make-deb

Installed module on BRM dev server

```
eecsaruh2hos055% pip install schedule-1.1.0-py2.py3-none-any.whl -f ./ --no-index --no-deps
Defaulting to user installation because normal site-packages is not writeable
Looking in links: ./
Processing ./schedule-1.1.0-py2.py3-none-any.whl
Installing collected packages: schedule
Successfully installed schedule-1.1.0
eecsaruh2hos055% python
Python 3.7.10 (default, Jul  7 2021, 06:41:22)
[GCC 10.3.0] on sunos5
Type "help", "copyright", "credits" or "license" for more information.
>>> import schedule
>>> exit()
```

Offline Uninstall module with pip

```
eecsaruh2hos055% pip uninstall schedule
Found existing installation: schedule 1.1.0
Uninstalling schedule-1.1.0:
  Would remove:
    /export/home/billdev/.local/lib/python3.7/site-packages/schedule-1.1.0.dist-info/*
    /export/home/billdev/.local/lib/python3.7/site-packages/schedule/*
Proceed (y/n)? y
```

Successfully uninstalled schedule-1.1.0

eecsaruh2hos055%

schedule in python program

pip install schedule

```
import schedule
import time

def job():
    print("I'm working...")

schedule.every(10).minutes.do(job)
schedule.every().hour.do(job)
schedule.every().day.at("10:30").do(job)

or
schedule.every()
  .day.at("10:30")
  .do(job) # # # Everyday at 10: 30
#schedule.every(5)
  .seconds.do(job) # # # Every 5 seconds
#schedule.every()
  .sunday.at("10:30")
  .do(job) # # # Every Sunday at 10: 30

while 1:
    schedule.run_pending()
    time.sleep(1)
```

Just run the job, it will run in foreground for given time.

Install modules from requirement.txt

pip install -r requirements.txt

To install / upgrade the modules with upgrade

pip install --upgrade pandas
(or)
python --m pip install --upgrade pandas

To upgrade a specific higher version

Pip install pandas==specific-higher-version

Managing Your Python Project with Git and PyBuilder

Installation

PIP : sudo pip install pybuilder

Installing From Source

pip install git+git://github.com/pybuilder/pybuilder

Building From Source

git clone https://github.com/pybuilder/pybuilder
cd pybuilder

To build simply run

./build.py
Congratulations, you just built PyBuilder from source!

CODES/Section 1:

5.Document-to-install-and-use-atom-editor.txt

```
open browser
        enter url: atom.io
        download option --> download atom , it is a .exe
        then double click on that, and wait some time.

install "script" package on your atom.

go to --> File --> settings --> install --> search with script , scirpt
package --> install it


and use ctrl+shift+b to run code
```

CODES/Section 10:

6.Document-search-a-file.py

```
import os
import string
import platform
req_file=input("Enter your file name to search: ")
```

```
if platform.system()=="Windows":
        pd_names=string.ascii_uppercase
        vd_names=[]
        for each_drive in pd_names:
                if os.path.exists(each_drive+":\\"):
                        #print(each_drive)
                        vd_names.append(each_drive+":\\")
        print(vd_names)
        for each_drive in vd_names:
                for r,d,f in os.walk(each_drive):
                        for each_f in f:
                                if each_f==req_file:
                                        print(os.path.join(r,each_f))
else:
        for r,d,f in os.walk("/"):
                for each_file in f:
                        if each_file==req_file:
                                print(os.path.join(r,each_file))
```

CODES/Section 11:

1.Document-check-given-path-is-file-dir-path.py

```
#This is a platform independet script(you can run on any operating system)
import os
path=input("Enter your path: ")

if os.path.exists(path):
        print(f"Given path : {path}is a valid path")
        if os.path.isfile(path):
                print("and it is a file path")
        else:
                print("and it is a directory path")
else:
        print(f"Given path : {path} is not existing on this host")
```

2.Document-read-a-path-and-identify-files-and-dirs.py

```
'''
import os
import sys
path=input("Enter your directory path: ")
if os.path.exists(path):
        df_l=os.listdir(path)
else:
        print("please provide valid path")
        sys.exit()


print(df_l)
p1=os.path.join(path,df_l[0])
p2=os.path.join(path,df_l[1])

if os.path.isfile(p1):
        print(f"{p1} is a file")
```

```
else:
        print(f"{p1} is a directory")

if os.path.isfile(p2):
        print(f"{p2} is a file")
else:
        print(f"{p2} is a directory")
'''



'''
print("befor loop")

for each in [2,3,4,5]:
        print("hello",each)


print("after loop")
'''

import os
import sys
path=input("Enter your directory path: ")
if os.path.exists(path):
        df_l=os.listdir(path)
else:
        print("please provide valid path")
        sys.exit()


list_of_files_dir=os.listdir(path)
print("all files and dirs: ",list_of_files_dir)
for each_file_or_dir in list_of_files_dir:
        f_d_p=os.path.join(path,each_file_or_dir)
        if os.path.isfile(f_d_p):
                print(f'{f_d_p} is a file')
        else:
                print(f'{f_d_p} is a directory')
```

4.Document-read-string-print-with-index-values.py

```
usr_str=input("Enter your string: ")

index=0
for each_char in usr_str:
        print(f'{each_char} -->{index}')
        index=index+1
```

5.Document-find-all-files.py*

```
#!/usr/local/bin/python3
import os
req_path=input("Enter your directory path: ")
#req_ex=input("Enter the required files extention .py/.sh/.log/.txt: ")
```

```
if os.path.isfile(req_path):
  print(f"The given path {req_path} is a file. Please pass only directory
path")
else:
  all_f_ds=os.listdir(req_path)
  if len(all_f_ds)==0:
     print(f"The given path is {req_path} an empty path")
  else:
     req_ex=input("Enter the required files extention .py/.sh/.log/.txt: ")
     req_files=[]
     for each_f in all_f_ds:
        if each_f.endswith(req_ex):
           req_files.append(each_f)
     if len(req_files)==0:
        print(f"There are no {req_ex} files in the logcation of {req_path}")
     else:
        print(f"There are {len(req_files)} files in the location of
{req_path} with an extention of {req_ex}")
        print(f"So, the files are: {req_files}")
```

8.Document-introduction-to-while.py

```
'''
while True:
       print("Welcome to loops")
       print(".........................")
'''
'''
import time
while True:
       print("MOnitoring file system usage")
       time.sleep(1)
'''


'''
value=4
while value<=6789:
       print(value)
       value=value+456
'''

'''
cnt=1
while cnt <=5:
       print("hello")
       cnt=cnt+1

'''
```

9.Document-control-statements.py

```
'''
for each in [3,4,56,7,8]:
       print(each)
       if each==56:
```

```
                break


print("after loop")
'''
'''
paths=['/usr/bin','/usr/bin/httpd','/home/users/xyz/weblogic/config.xml']
for each_path in paths:
        print("now working on: ",each_path)
        if 'httpd' in each_path:
                print(each_path)
                break
print("outside of for loop")
'''

'''
cnt=1
while True:
        print(cnt)
        if cnt==100:
                break
        cnt=cnt+1

'''


'''

for each in range(1,11):
        if each ==7:
                continue
                print("this is the line inside of your if condition after
continue keyword")
        print(each)
'''


'''
if True:
        pass
'''
'''
for each in range(3):
        pass
'''
```

CODES/Section 12:

2.Document-delete-all-files-which-are-older-than-x-days.py

```
import os
import sys
import datetime
req_path=input("Enter your path: ")
age=3
if not os.path.exists(req_path):
   print("Please provide valid path ")
```

```
   sys.exit(1)
if os.path.isfile(req_path):
   print("Please provide directory path ")
   sys.exit(2)
today=datetime.datetime.now()
for each_file in os.listdir(req_path):
   each_file_path=os.path.join(req_path,each_file)
   if os.path.isfile(each_file_path):

file_cre_date=datetime.datetime.fromtimestamp(os.path.getctime(each_file_path
))
      #print(dir(today-file_cre_date))
      dif_days=(today-file_cre_date).days
      if dif_days > age:
         print(each_file_path,dif_days)
```

CODES/Section 13:

1.Document-introduction-to-subprocess.txt

```
import subprocess
sp=subprocess.Popen(cmd,shell=True/False,stdout=subprocess.PIPE,stderr=subpro
cess.PIPE,universal_newlines=True)
rc=sp.wait()
out,err=sp.communicate()
print(f'OUTPUT IS: {out}')
print(f'Error is: {err}')
================================>
if shell=True then your cmd is a string (as your os command)
if shell=False then your cmd is a list

Note: shell=False dont work on your os environment variables

ex: cmd="ls -lrt" ==>shell=True
    shell=False ==> cmd="ls -lrt".split()  or ['ls','-lrt']
=========================================================================


shell=True always on windows
============================
cmd is a string
-------------------------------------------------------------
```

2.Document-find-java.py

```
import subprocess
cmd="java -version"
sp=subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.P
IPE,universal_newlines=True)
rc=sp.wait()
o,e=sp.communicate()
if rc==0:
    if bool(o)==True:
        print(o)
    #print(bool(o),bool(e))
```

```
    '''
    if bool(o)==False and bool(e)==True:
        print(e.splitlines()[0].split()[2].strip("\""))
    '''
    if bool(o)==False:
        if bool(e)==True:
            print(e.splitlines()[0].split()[2].strip("\""))
else:
    print(e)
```

3.Document-get-bash-version.py

```
import subprocess
cmd=["bash","--version"]
sp=subprocess.Popen(cmd,shell=False,stdout=subprocess.PIPE,stderr=subprocess.
PIPE,universal_newlines=True)
rc=sp.wait()
o,e=sp.communicate()

if rc==0:
   for each_line in o.splitlines():
      if "version" in each_line  and "release" in each_line:
         print(each_line.split()[3].split('(')[0])
else:
   print("Command was failed and error is: ",e)
```

CODES/Section 14:

1.Document-create-a-file.py

```
#Working with text files
'''
fo=open('newdemo.txt','w')
#print(fo.mode)
#print(fo.readable())
#print(fo.writable())
fo.close()
'''
'''
my_content=["This is a data -1\n","This is a data-2\n","This is a data-3"]
fo=open("random.txt",'w')
fo.write("This is a first line\n")
fo.write("This is a second line\n")
fo.write("This is a third line")
#fo.writelines(my_content)
fo.close()
'''
'''
my_content=['This is using loop-iteratioin-1','This is using loop-iterantion-
2','This is using loop-iteratioin-3']

fo=open("with_loop.txt",'a')

for each_line in my_content:
        fo.write(each_line+"\n")
fo.close()
'''
```

```
'''

fo=open("with_loop.txt","r")
data=fo.read()
fo.close()

print(data)
'''
'''
fo=open("with_loop.txt","r")
print(fo.readline())
print(fo.readline())
fo.close()
'''

fo=open("with_loop.txt","r")
data=fo.readlines()
fo.close()
'''
for each in range(3):
        print(data[each])    #data[0], data[1],data[2]
'''
print(data[-1])
```

2.Document-s-d.py

```
#sfile="C:\\Users\\Automation\\Desktop\\random.txt"
#dfile="C:\\Users\\Automation\\Downloads\\newrandom.txt"
sfile=input("Enter your source file: ")
dfile=input("Enter your destination file: ")
sfo=open(sfile,'r')
content=sfo.read()
sfo.close()

dfo=open(dfile,'w')
dfo.write(content)
dfo.close()
```

CODES/Section 15:

1.Document-read-csv-file.py

```
import csv
req_file="C:\\Users\\Automation\\Desktop\\hi\\new_info.csv"

fo=open(req_file,"r")
content=csv.reader(fo,delimiter="|")
for each in content:
        print(each)

fo.close()
```

2.Document-read-a-header-and-finding-no-of-rows.py

```
import csv
req_file="C:\\Users\\Automation\\Desktop\\hi\\new_info.csv"
```

```
fo=open(req_file,"r")
content=csv.reader(fo,delimiter="|")
#print(list(content))
#print(f'The header is:\n {list(content)[0]}')
#header=next(content)
#print("The header is: ",header)
print("The no of rows are: ",len(list(content))-1)
'''
for each in content:
        print(each)
'''
fo.close()
```

3.Document-create-csv-file.py

```
import csv
#req_file="C:\\Users\\Automation\\Desktop\\hi\\new_info.csv"
'''
fo=open(req_file,'r')
csv_reader=csv.reader(fo,delimiter="|")
for each_row in csv_reader:
        print(each_row)
fo.close()
'''
req_file="C:\\Users\\Automation\\Desktop\\hi\\demo.csv"
fo=open(req_file,'w',newline="")
csv_writer=csv.writer(fo,delimiter=",")
'''
csv_writer.writerow(['S_No',"Name",'Age'])
csv_writer.writerow([1,"John",23])
csv_writer.writerow([2,"Cliton",24])
'''
my_data=[['S_No',"Name",'Age'],[1,"John",23],[2,"Cliton",24]]
csv_writer.writerows(my_data)
fo.close()
```

CODES/Section 16:

working-with-json-files.py

```
import json
#Read a json file
'''
req_file="myjson.json"

fo=open(req_file,'r')
#print(fo.read())
print(json.load(fo))

fo.close()
'''
#Write data(dictionary data) into a json file
my_dict={'Name':'Narendra','skills':['Python','shell','yaml','AWS']}
```

```
req_file="myinfo.json"

fo=open(req_file,'w')
json.dump(my_dict,fo,indent=4)

fo.close()
```

CODES/Section 17:

1.Document-.working-with-exceptions.py

```
'''
print("Welcome to exceptions concept")
print("Now it is fine")
fo=open("nari.txt")
'''

'''
try:
        fo=open("nari.txt")
        print(fo.read())
        fo.close()
except Exception as e:
        print(e)
        '''


my_list=[3,4,5]


'''

try:
        print(my_list[4])
except Exception as e:
        print(e)
'''

#print(my_list[4])


#print("This code will also execetues")


try:
        import fabric
except Exception as e:
        print(e)
```

2.Document-exception-handling-for-known-exceptions.py

```
#NameError
#TypeError
#FileNotFoundError
```

```
#ZeroDivisionError

try:
        print("This is try block")
        import fabric
        print(a)
        #print(4+"hi")
        #open('asdfas.txt')
        #print(5/0)

except FileNotFoundError:
        print("File is not present to open it")
except NameError:
        print("Variable is not defined")
except TypeError:
        print("Adding number and string is not possible")
except ZeroDivisionError:
        print("Division with zero is not possible")
except ModuleNotFoundError:
        print("Please install fabric to use it")
except Exception as e:
        print(e)
finally:
        print("Finally this will executes")
```

3.Document-differce-finally-esle.py

```
try:
        a=9
        print(a)
except NameError:
        print("Variable is not defined")
except Exception as e:
        print("Exception occured:",e)
else:
        print("This will execute if there is no exceptions in try block")
finally:
        print("This will executes always")
```

4.Document-custom-exceptions.py

```
'''

age=23


if age>30:
        print("Valid age")
else:
        raise ValueError("Age is less than 30")

'''
age=20

try:
        assert age>30
```

```
        print("Valid age")
except AssertionError:
        print("Raised with assert because age is lessthan 30")
except:
        print("Exception occured")
```

CODES/Section 18:

1.Document-run-your-commands.py

```
import os
import time
import platform
def mycode(cmd1,cmd2):
        print("Please wait. Cleaning the screen....")
        time.sleep(2)
        os.system(cmd1)
        print("Please wait finding the list of dir and files")
        time.sleep(2)
        os.system(cmd2)
if platform.system()=="Windows":
        mycode("cls","dir")
else:
        mycode('clear','ls -lrt')
```

10.variable-leng-keyword.py

```
def display(p,**karg):
        print(p)
        print(karg)
        return None
#display(4,5)
#display(b=5,a=4)
#display(a=4,b=5,c=6)
display(56,x=5,y="Hi",z=6.7,user="root")
```

12.Document-change-working-dir.py*

```
#!/usr/local/bin/python3
import os
'''
req_path=input("Enter path to change working dir: ")
print("The current working dir is: ",os.getcwd())
try:
  os.chdir(req_path)
  print("Now your new working dir is: ",os.getcwd())
except FileNotFoundError:
  print("Given path is not a valid path. So cant change working directory")
except NotADirectoryError:
  print("Given path is a file path. So cant change working directory")
except PermissionError:
  print("Sorry you dont have access for the given path. So cant chagne
working directory")
except Exception as e:
  print(e)
```

```
'''

def main():
  req_path=input("Enter path to change working dir: ")
  print("The current working dir is: ",os.getcwd())
  try:
    os.chdir(req_path)
    print("Now your new working dir is: ",os.getcwd())
  except FileNotFoundError:
    print("Given path is not a valid path. So cant change working directory")
  except NotADirectoryError:
    print("Given path is a file path. So cant change working directory")
  except PermissionError:
    print("Sorry you dont have access for the given path. So cant chagne
working directory")
  except Exception as e:
    print(e)
  return None


if __name__=="__main__":
   main()
```

2.Document-defining-fun.py

```
def display():
        print("Welcome to functions concept")
        print("Simple way to define your function")
        return None

display()


print(len("hi"))
x=40
print(id(x))


#Rules to define function name:
#Function name should have only a-z,A-Z,0-9, _
#Function shuold not start with number but can it be start with _
#Dont include any space.
#Function must be define befor calling it


x=(5,6)
print(len(x))

print(min(x))
print(max(x))



x="5"
print(int(x))
```

3.Documnet-converting-into-fun.py.py

```
def welcome_msg():
        print("Welcome to Python Scripting")
        print("Python is easy to learn")
        return None
def known_concepts():
        print("Now we are good with bascis")
        print("We are about to start functions concepts in python")
        return None
def learning_concepts():
        print("Function are very easy in python")
        print("Now we are writing simple functions")
        return None
welcome_msg()
known_concepts()
learning_concepts()
```

4.Document-scope-of-the-variables.py

```
def myfunction1():
        x=60  #This is local variable
        print("Welcome to functions")
        print("x value from fun1: ",x)
        #myfunction2()
        return None


def myfunction2(y):  #Parameter
        print("Thank you!!")
        print("x value from fun2: ",y)
        return None

def main():
        #global x
        x=10
        myfunction1()
        myfunction2(x)  #Argument
        return None


main()
```

5.Document-function-with-simple-arguments.py

```
'''
def get_result(value):  #parameters/positional arguments
        result=value+10
        print(f'Your result is: {result}')
        return None
def main():
        #global num
        num=eval(input("Enter your number: "))
        get_result(num)  #Arguments
        return None
```

```
main()
'''
def get_add(p,q):
        aresult=p+q
        print(f'The addition of {p} and {q} is: {aresult}')
        return None
def get_sub(m,n):
        sresult=m-n
        print(f'The sub of {m} and {n} is: {sresult}')
        return None

def main():
        a=eval(input("Enter your first num: "))
        b=eval(input("Enter your second num: "))
        get_add(a,b)
        get_sub(a,b)
        #x=50
        #get_sub(19,x)
        return None

main()
```

6.Document-fun-with-arg-return-values.py

```
'''
def get_addition(a,b):
        result=a+b
        #print(f"The addition of {a} and {b} is: {result}")
        return result
def main():
        a=eval(input("Enter your first number: "))
        b=eval(input("Enter your second number :"))
        result=get_addition(a,b)
        print(f"The addition of {a} and {b} is: {result}")
        return None
main()
'''

def multiply_num_10(value):
        #result=value*10
        #return result
        return value*10


def main():
        num=eval(input("Enter your number: "))
        result=multiply_num_10(num)
        print("The result is: ",result)



main()
```

7.Document-fun-with-default-arguments.py

```
'''
```

```
def display(a=1):
        print("The value of a is: ",a)
        return None

display(4)
display(5)
display()
'''

'''
def add_numbers(a,b=0):
        result=a+b
        print("The result is: ",result)
        return None
add_numbers(4,5)
add_numbers(5)
add_numbers(7)
'''

def working_on_some(user="root"):
        print(f"working with {user}")
        return None

working_on_some("weblogic_admin")
```

8.Document-function-with-keyword-based-arguments.py

```
def display(a,b):
        print(f'a={a}')
        return None

display(3,4)
display(a=3,b=4)
display(b=4,a=3)
```

9.Document-functions-with-variable-length-argument.py

```
def display(*arg):
        for each in arg:
                print(type(each))
        return None
#display()
#display(4)
display(4,5,67)
print('-------------')
display("hi",4.65)
```

CODES/Section 19:

2.Document-0regexe-pat-rules-creation.py

```
import re
text="This . is a python and it is easy to learn and it is popular one for
dev and automation"
'''
```

```
my_pat= 'i[ston]' #is,it,io,in
#print(len(re.findall(my_pat,text)))
print(re.findall(my_pat,text))

'''


#my_pat="x[abcdeflmnopq]y" ==> xay,xby.....xqy
#      ="x[a-fl-q]y"
'''
my_pat='\.'
print(re.findall(my_pat,text))
'''
'''
text="This is a ip address of my db1 server: 255.255.255.255
2456234512341234"

my_pat="\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d"
print(re.findall(my_pat,text))
'''


'''
text="This is python @ 345 _ - ("
print(re.findall('\w',text))
#print(re.findall('.',text))
print(re.findall("\W",text))
'''


text="456 90 this is about deciaml re98gex"
print(re.findall('\d\d',text))
```

3.Document-rules-2.py

```
import re
text='isa python l     earn and \n itis easy  to'
#my_pat='^i[ts]'
#my_pat="learn$"
#my_pat=r"\blearn\b"
#my_pat=r"\Blearn\B"
my_pat=r"\n"
print(re.findall(my_pat,text))
```

4.Document-rules-3.py

```
import re
'''
text="This is a pythonnnn and python aaa haaaafd xyzaaaaaaaa"
#my_pat=r'\bpython{4}\b'
my_pat=r'\bxyza{8}\b'
print(re.findall(my_pat,text))
'''
text="xaz asdfa sdf xaaz xaaaaaaaz xaaaaz  xz"
#my_pat=r'\bxa{2,}z\b'
#my_pat=r'xa{1,}z'
```

```
#my_pat=r'xa*z'
my_pat=r'xa?z'
print(re.findall(my_pat,text))
```

5.Document-regex-flags.py

```
import re
'''
text="this is a string ThIs is a new staring THIS"
my_pat=r'this'
print(re.findall(my_pat,text,re.I))
'''
text="""this is a string EnD
this is second line enD
This is third line end
asfasd this end"""
#print(text)

#my_pat=r'^this'
my_pat=r'end$'

print(re.findall(my_pat,text,re.M|re.I))
```

6.Document-search-and-match-practice.py

```
import re
text="""This  is for
python2 and there are two major
vers python3 and python in future python4"""

pat=r'\bpython[23]?\b'
'''
#print(re.findall(pat,text))
match_ob=re.search(pat,text)
#rint(match_ob)
if match_ob:
        print("match from ur pattern: ",match_ob.group())
        print('Starting index: ',match_ob.start())
        print('Ending index: ',match_ob.end()-1)
        print("Length: ",match_ob.end()-match_ob.start())
else:
        print("No match found")
'''
text="""PYTHON2 and there are two major
vers python3 and python in future python4"""

pat=r'\bpython[23]?\b'
match_ob=re.match(pat,text,re.I)
if match_ob:
        print("match from ur pattern: ",match_ob.group())
        print('Starting index: ',match_ob.start())
        print('Ending index: ',match_ob.end()-1)
        print("Length: ",match_ob.end()-match_ob.start())
else:
        print("No match found")
```

7.Document-find-all-find-iter.py

```
import re
my_str="This is python and we are having python2 and python3 version"
my_pat=r'\bpython[23]?\b'
#print(re.search(my_pat,my_str))
#print(len(re.findall(my_pat,my_str)))
#print(re.findall(my_pat,my_str))

for each_ob in re.finditer(my_pat,my_str):
        print(f'The match is: {each_ob.group()} starting index:
{each_ob.start()}, ending index {each_ob.end()-1}')
```

9.Document-complie-operations.py

```
import re
my_str="This is about python. Python is easy to learn  and we have two major
versions: python2 and python3 "

my_pat=r'\bPython[23]?\b'

#print(re.search(my_pat,my_str))
#print(re.findall(my_pat,my_str,flags=re.I))
#print(re.split(my_pat,my_str))


pat_ob=re.compile(my_pat,flags=re.I)
print(pat_ob)
print(pat_ob.search(my_str))
print(pat_ob.findall(my_str))


#re.findall(my_pat,my_str)===> re.complie(my_pat).findall(my_str)
```

CODES/Section 20:

1.Document-working-with-remote-linux-from-windows.py*

```
#!/bin/python
import paramiko
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
#ssh.connect(hostname='3.92.79.119',username='ec2-
user',password='paramiko123',port=22)
ssh.connect(hostname='3.92.79.119',username='ec2-
user',key_filename='/home/Automation/.ssh/id_rsa',port=22)
#stdin, stdout, stderr = ssh.exec_command('whoami')
#stdin, stdout, stderr = ssh.exec_command('uptime')
stdin, stdout, stderr = ssh.exec_command('free -m')
print("The output is: ")
print(stdout.readlines())


print("THe error is: ")
print(stderr.readlines())
```

2.Document-transfer_files.py

```
import paramiko
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect(hostname='54.165.97.91',username='ec2-
user',password='paramiko123',port=22)
sftp_client=ssh.open_sftp()

#sftp_client.get('/home/ec2-
user/paramiko_download.txt','paramiko_downloaded_file.txt')
#sftp_client.chdir("/home/ec2-user")
#print(sftp_client.getcwd())
#sftp_client.get('demo.txt','C:\\Users\\Automation\\Desktop\\download_file.tx
t')
sftp_client.put("transfer_files.py",'/home/ec2-user/transfer_files.py')
sftp_client.close()
ssh.close()
```

CODES/Section 21:

shutil-part-1.py*

```
#!/usr/local/bin/python3
import shutil
#copy', 'copy2', 'copyfile', 'copyfileobj', 'copymode', 'copystat',
'copytree'

#copyfile-->copy --> copy2
#src="/home/Automation/working_with_remote_server.py"
src="/home/Automation/shutil_part_1.py"
dest="/home/Automation/working_with_remote_server.py_bkp"
#shutil.copyfile(src,dest)
#shutil.copy(src,dest)   #same permissions for src and dest
#shutil.copy2(src,dest)   #same meta data for dest as well
#shutil.copymode(src,dest)
#shutil.copystat(src,dest)

#f1=open('xyz.txt','r')
#f2=open('pqr.txt','w')
#shutil.copyfileobj(f1,f2)

#src="/home/Automation/tomcat7"
#shutil.copytree(src,'/tmp/tomcat')

shutil.rmtree('/tmp/tomcat')
```

CODES/Section 22:

1.Document-introduction_to_oops.py

```
import os
class Tomcat:
        def get_details_for_each_tomcat(self,server_xml):
                self.tcf=server_xml
```

```
                self.th=os.path.dirname(os.path.dirname(server_xml))
                return None

        def display_details(self):
                print(f'The tomcat config file is: {self.tcf}\nThe tomcat home
is: {self.th}')
                return None

def main():
        tomcat7=Tomcat()
        tomcat9=Tomcat()

        tomcat7.get_details_for_each_tomcat("/home/Automation/tomcat7/conf/ser
ver.xml")
        #get_details_for_each_tomcat('tomcat7',"/home/Automation/tomcat7/conf/
server.xml")
        tomcat9.get_details_for_each_tomcat("/home/Automation/tomcat9/conf/ser
ver.xml")
        #get_details_for_each_tomcat('tomcat9',"/home/Automation/tomcat9/conf/
server.xml")
        #print(tomcat9.tcf)
        #print(tomcat7.th)
        #print(tomcat9.th)
        #print(tomcat7.tcf)
        tomcat9.display_details()
        tomcat7.display_details()
        #display_details('tomcat7')
        tomcat9.display_details()

        return None

if __name__=="__main__":
        main()
```

2.Document-working_on_attributes.py

```
class emp:
        count=0
        def get_name_age_salary(self,name,age,salary):
                self.name=name
                self.age=age
                self.salary=salary
                self.increase_count_for_emp()
                #self.display_details()
                return None
        def increase_count_for_emp(self):
                emp.count=emp.count+1
                return None
        def display_details(self):
                print(f'The name is: {self.name}\nThe age is: {self.age}\nThe
salary is: {self.salary}')
                return None

emp1=emp()
emp2=emp()
```

```
emp1.get_name_age_salary('John',34,45000)
#emp1.increase_count_for_emp()
emp2.get_name_age_salary('Cliton',25,54000)
#emp2.increase_count_for_emp()

print(emp.count)
```

3.Document-using_init_method.py

```
class Emp(object):
        def __init__(self,name,salary):
                self.name=name
                self.salary=salary
                return None
        def display(self):
                print(f"The name is: {self.name}\nThe salary is:
{self.salary}")
                return None


emp1=Emp('Ramu',56000)
emp2=Emp("Naren",90000)

emp1.display()
#emp2.display()
```

4.1.Document-without_oops.py*

```
#!/bin/python3
import os

def get_all_tomcats():
    list_of_config_files=[]
    for r,d,f in os.walk("/"):
        for each_file in f:
            if each_file=='server.xml':
                list_of_config_files.append(os.path.join(r,each_file))
    return list_of_config_files
def display_details(home_cnf_files):
    for each_key in home_cnf_files.keys():
        print(f'The Tomcat home is: {home_cnf_files[each_key][0]}')
        print(f'The Tomcat config file is:{home_cnf_files[each_key][1]}' )
    return None

def main():
    print("Finding list of tomcats...")
    list_of_tomcats=get_all_tomcats()
    #print(list_of_tomcats)
    cnt=1
    home_cnf_files={}
    for each_config_file in list_of_tomcats:
        t_home=os.path.dirname(os.path.dirname(each_config_file))
        t_cnf_file=each_config_file
        home_cnf_files['tomcat'+str(cnt)]=[t_home,t_cnf_file]
        cnt+=1
```

```
    #print(home_cnf_files)
    display_details(home_cnf_files)
    return None


if __name__ == '__main__':
    main()
```

4.2.Document-working_with_oops.py

```
class emp:
        count=1


emp.count=emp.count+1
print(emp.count)
```

5.Document-constructor_and_destructor_of_an_object.py

```
class Person(object):
        def __init__(self,name,age):
                print("an object has been created")
                self.name=name
                self.age=age
                return None
        def __del__(self):
                print("object has been deleted")
                return None


per1=Person('Jhon',26)
```

6.1.Document-polymorphism.py

```
class Tomcat(object):
        def __init__(self,home,ver):
                self.home=home
                self.version=ver
                return None
        def display(self):
                print("This is from tocmat class")
                print(self.home)
                print(self.version)
                return None
class Apache(object):
        def __init__(self,home,ver):
                self.home=home
                self.version=ver
                return None
        def display(self):
                print("This is from apache class")
                print(self.home)
                print(self.version)
                return None
```

```
tom_ob=Tomcat('/home/tomcat9','7.6')
apa_ob=Apache("/etc/httpd",'2.4')

tom_ob.display()
apa_ob.display()
```

6.2.Document-inheritance.py

```
class Tomcat(object):
        def __init__(self,home,ver):
                self.home=home
                self.version=ver
                return None
        def display(self):
                print(self.home)
                print(self.version)
                return None
class Apache(Tomcat):
        def __init__(self,home,ver):
                self.home=home
                self.version=ver
                return None
tom_ob=Tomcat('/home/tomcat9','7.6')
apa_ob=Apache("/etc/httpd",'2.4')


tom_ob.display()
apa_ob.display()
```

7.Document-Encpasulation.py

```
class Person(object):
        def assing_name_and_age(self,name,age):
                self.name=name
                self.__age=age
                self.__display()
                return None
        def __display(self):
                print(self.name,self.__age)
                return None

per1=Person()
per1.assing_name_and_age('John',32)

#per1.__display()
#print(per1.name)
#print(per1.__age)
```

Python practice programs

https://www.codesdope.com/practice/python-go-for-and-for/

```
'''Take inputs from user to make a list.
Again take one input from user and search it in the list and delete that
element, if found.
Iterate over list using for loop.'''

#Take inputs from user to make a list.
usr_in = input("Please Enter your Input string:")
print(f"You Entered string is: {usr_in}")

usr2_in = input("Please search a string from above to delete:")
print(f"You are searching for: {usr2_in}")

usr3_in = ""
# if usr2_in in usr_in:
usr_in.find(usr2_in)
print(usr_in.find(usr2_in))
print(f"===={usr_in}====")
    # print(f"{usr_in.split(usr2_in)}")
usr1_in = str(usr_in.split(usr2_in))
print(usr1_in)
    # print(usr3_in)
listToStr = ''.join([str(elem) for elem in usr1_in])
print(listToStr)
print(type(listToStr))
```

```
'''Print multiplication table of 14 from a list in which multiplication table
of 12 is stored.'''

i = "14"
j = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for jno in j:
    print(f"{i}*{jno} = {int(i) * int(jno)}")
```

```
'''Print multiplication table of 14 from a list in which multiplication table
of 12 is stored.'''

i = input("Please Enter the table no:")
j = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for jno in j:
    print(f"{i}*{jno} = {int(i) * int(jno)}")
```

```
'''You are given with a list of integer elements.
Make a new list which will store square of elements of previous list.'''

i = [2,5,2,7,1,9]
for a in i:
    print(f"{a**2}")
```

```
'''Using range(1,101), make two list,
one containing all even numbers and other containing all odd numbers.'''

even = []
odd = []
```

```
j = 0
for i in range(1, 101):
    # print(i)
    # print(type(i))
    if i % 2 == 0:
        even.append(i)

    else:
        odd.append(i)


print(f"even list: \n{even}")
print(f"odd list: \n{odd}")
```

https://www.section.io/engineering-education/flask-database-integration-with-sqlalchemy/

Important modules in python

```
import os
import time
import platform
import pandas as pd
import requests
import re
import paramiko
import sqlite3
import flask_sqlalchemy
import sqlalchemy
import shutil
import psutil
import rpa
```

Google client API's

https://developers.google.com/api-client-library/

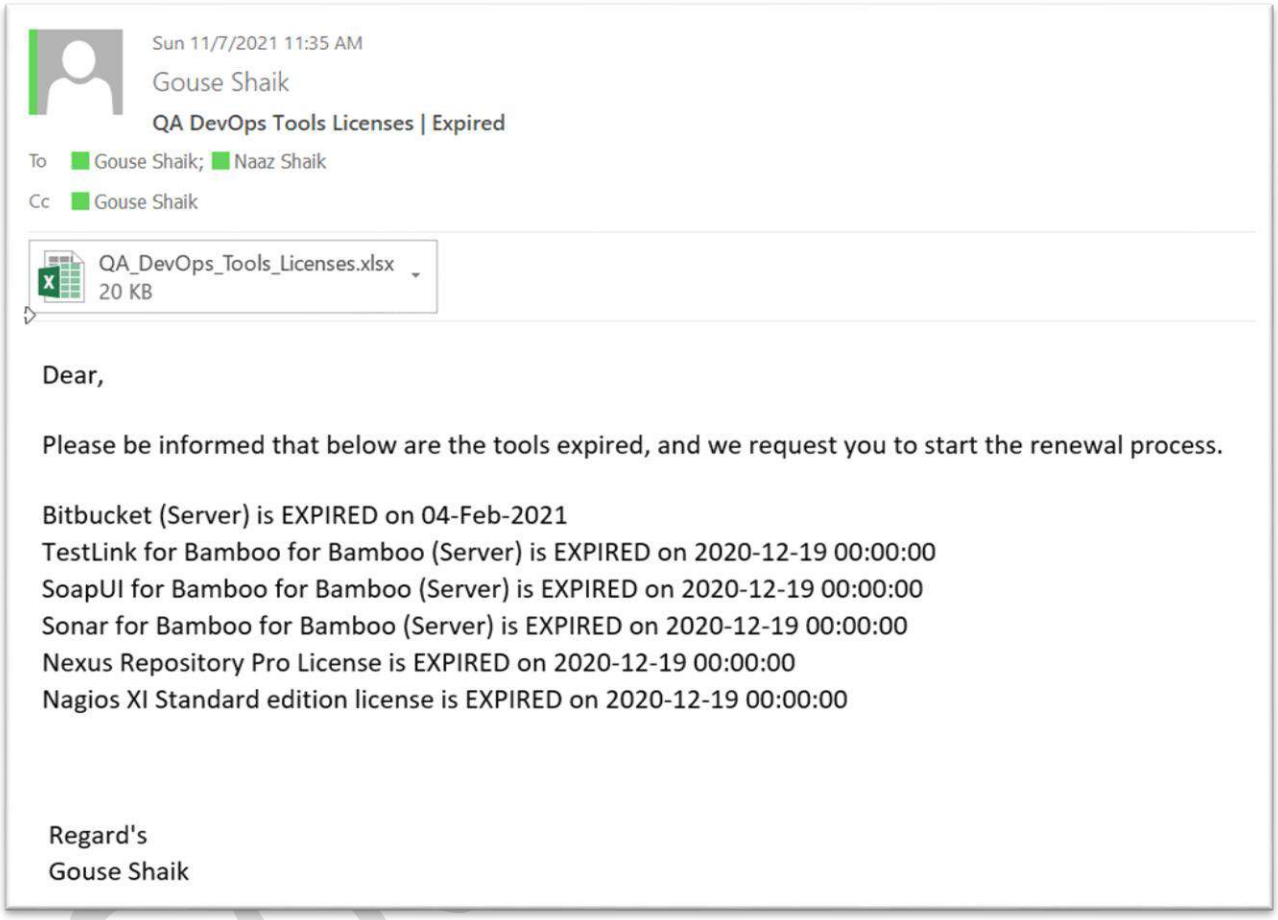
Send tools licenses expired notification to licenses renewal code.

```
import openpyxl
import win32com.client


# def send_mail():
# https://www.codeforests.com/2020/06/05/how-to-send-email-from-outlook/
# initiate the outlook application
# outlook = win32com.client.Dispatch('outlook.application')
# In outlook, email, meeting invite, calendar, appointment etc.
# are all considered as Item object. Hence we can use the below to create an
email object:
# mail = outlook.CreateItem(0)
# for this mail item, there are various attributes we can set,
# such as the below To, CC, BCC, Subject, Body, HTMLBody etc. as well as the
Attachments:
# mail.To = 'g.shaik@mobily.com.sa'
# mail.Subject = 'Sample Email'
```

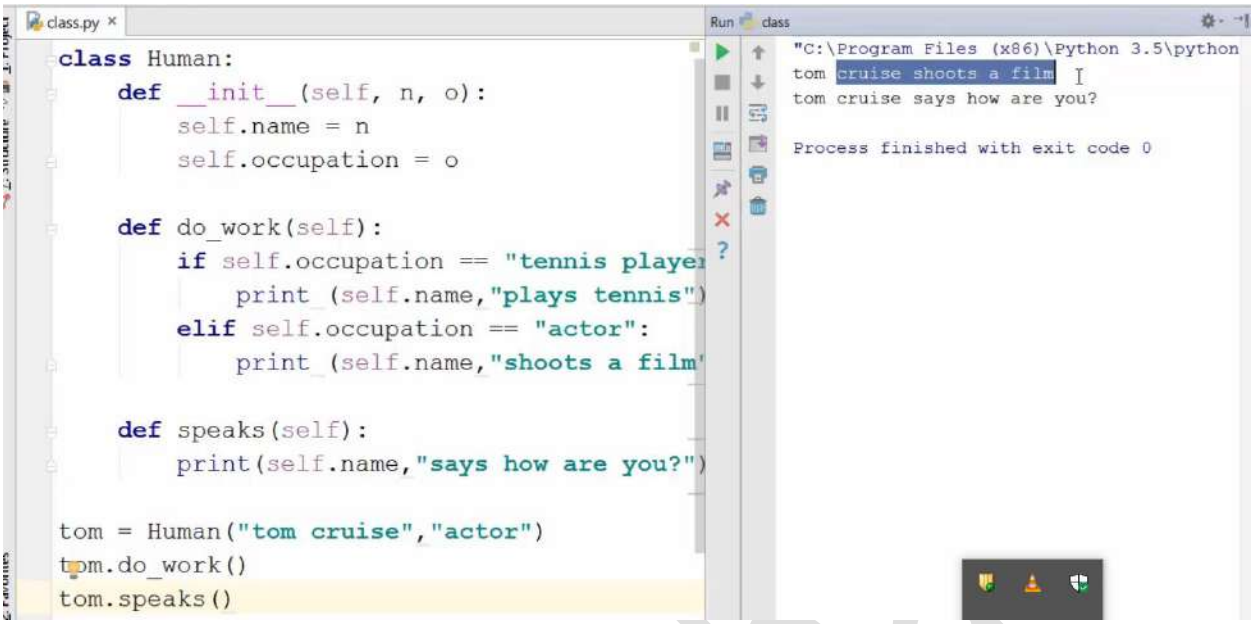
```
# mail.HTMLBody = '<h3>This is HTML Body</h3>'
# mail.Body = "This is the normal body"
# path = "Input\\QA_DevOps_Tools_Licenses.xlsx"
# mail.Attachments.Add(path)
# mail.Attachments.Add(path)
# mail.CC = 'somebody@company.com'
# Trigger to send out email from outlook
# mail.Send()

def renewal_status():
    # Location of the file
    # path = "Input\\QA_DevOps_Tools_Licenses.xlsx"
    path =
"C:\\Py_Projects\\QA_DevOps_Tools_Licenses\\Input\\QA_DevOps_Tools_Licenses.x
lsx"
    # with open(path, 'r') as myfile:
    #     data = myfile.read()

    # workbook object is created
    wb_obj = openpyxl.load_workbook(path)

    # Get workbook active sheet object
    # from the active attribute
    sheet_obj = wb_obj.active

    # Cell object is created by using
    # sheet object's cell() method.
    cell_obj = sheet_obj['C2': 'E9']

    # Print value of cell object
    # using the value attribute
    mail_body = ""
    for cell1, cell2, cell3 in cell_obj:
        if cell2.value == "EXPIRED":
            # new_value = f'{cell1.value}is {cell2.value} on {cell3.value}'
            new_value = str(cell1.value) + " is " + str(cell2.value) + " on "
+ str(cell3.value)
            new_value = new_value.replace(u'\xa0', u'')
            mail_body += new_value + '\n'
            # mail_body.append(new_value)
        else:
            pass
    outlook = win32com.client.Dispatch('outlook.application')
    mail = outlook.CreateItem(0)
    mail.To = 'g.shaik@mobily.com.sa; nshaik@mobily.com.sa'
    mail.Subject = 'QA DevOps Tools Licenses | Expired'
    mail.HTMLBody = '<h3>This is HTML Body</h3>'
    main_content = "Please be informed that below are the tools expired, and
we request you to start the renewal " \
                   "process. "
    mail_sign = " Regard's \n Gouse Shaik"
    mail.Body = f"Dear,\n\n{main_content}\n\n{mail_body}\n\n{mail_sign}"
    # path = "Input\\QA_DevOps_Tools_Licenses.xlsx"
    mail.Attachments.Add(path)
    # mail.Attachments.Add(path)
    mail.CC = 'g.shaik@mobily.com.sa'
    # Trigger to send out email from outlook
```

```
    mail.Send()


if __name__ == '__main__':
    renewal_status()
```

Output:



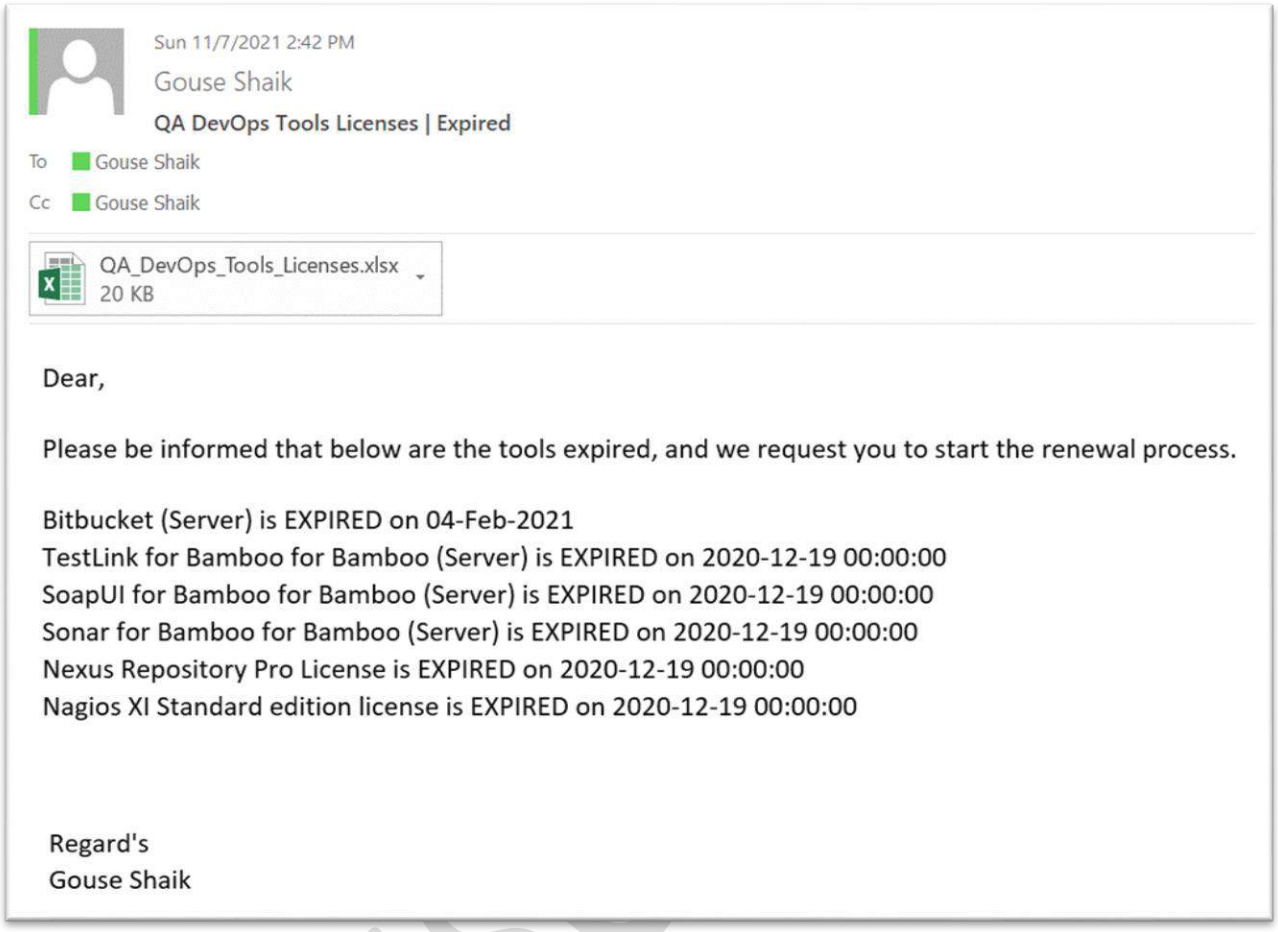
The same above DevOps Tools licenses python program in OOPs




```
import openpyxl
import win32com.client


# def send_mail():
# https://www.codeforests.com/2020/06/05/how-to-send-email-from-outlook/
# initiate the outlook application
# outlook = win32com.client.Dispatch('outlook.application')
# In outlook, email, meeting invite, calendar, appointment etc.
# are all considered as Item object. Hence we can use the below to create an
email object:
# mail = outlook.CreateItem(0)
# for this mail item, there are various attributes we can set,
# such as the below To, CC, BCC, Subject, Body, HTMLBody etc. as well as the
Attachments:
# mail.To = 'g.shaik@mobily.com.sa'
# mail.Subject = 'Sample Email'
# mail.HTMLBody = '<h3>This is HTML Body</h3>'
# mail.Body = "This is the normal body"
# path = "Input\\QA_DevOps_Tools_Licenses.xlsx"
# mail.Attachments.Add(path)
# mail.Attachments.Add(path)
# mail.CC = 'somebody@company.com'
# Trigger to send out email from outlook
# mail.Send()

class QATools:
    def __init__(self, excel_path):
        self.path = excel_path

    def renewal_status(self):
        # Location of the file
        # path = "Input\\QA_DevOps_Tools_Licenses.xlsx"
        # path =
"C:\\Py_Projects\\QA_DevOps_Tools_Licenses\\Input\\QA_DevOps_Tools_Licenses.x
```

```
lsx"
        # with open(path, 'r') as myfile:
        #     data = myfile.read()

        # workbook object is created
        wb_obj = openpyxl.load_workbook(self.path)

        # Get workbook active sheet object
        # from the active attribute
        sheet_obj = wb_obj.active

        # Cell object is created by using
        # sheet object's cell() method.
        cell_obj = sheet_obj['C2': 'E9']

        # Print value of cell object
        # using the value attribute
        mail_body = ""
        for cell1, cell2, cell3 in cell_obj:
            if cell2.value == "EXPIRED":
                # new_value = f'{cell1.value}is {cell2.value} on
{cell3.value}'
                new_value = str(cell1.value) + " is " + str(cell2.value) + "
on " + str(cell3.value)
                new_value = new_value.replace(u'\xa0', u'')
                mail_body += new_value + '\n'
                # mail_body.append(new_value)
            else:
                pass
        outlook = win32com.client.Dispatch('outlook.application')
        mail = outlook.CreateItem(0)
        mail.To = 'g.shaik@mobily.com.sa'
        mail.Subject = 'QA DevOps Tools Licenses | Expired'
        mail.HTMLBody = '<h3>This is HTML Body</h3>'
        main_content = "Please be informed that below are the tools expired,
and we request you to start the renewal " \
                       "process. "
        mail_sign = " Regard's \n Gouse Shaik"
        mail.Body = f"Dear,\n\n{main_content}\n\n{mail_body}\n\n{mail_sign}"
        # path = "Input\\QA_DevOps_Tools_Licenses.xlsx"
        mail.Attachments.Add(self.path)
        # mail.Attachments.Add(path)
        mail.CC = 'g.shaik@mobily.com.sa'
        # Trigger to send out email from outlook
        mail.Send()


# if __name__ == '__main__':
#     renewal_status()

devops_tools =
QATools("C:\\Py_Projects\\QA_DevOps_Tools_Licenses\\Input\\QA_DevOps_Tools_Li
censes.xlsx")
devops_tools.renewal_status()
```

output:



Get the full updated code from github:

Batch (.bat) schedule a python script

Open notepad

```
@py C:\Py_Projects\Pyflaskbootstrap\webautomate.py %*
@pause
```

Save with *webautomate.bat* and schedule with in windows scheduler.

JARVIS AI Assistant

1st module: pip install pyttsx3

In case you receive such errors:

- No module named win32com.client
- No module named win32
- No module named win32api

Then, install pypiwin32 by typing the below command in the terminal :

```
pip install pypiwin32. And then install pyttsx3
```

Text to Speech (TTS) library for Python 2 and 3. Works without internet connection or delay. Supports multiple TTS engines, including Sapi5, nsss, and espeak.

2nd module: pip install speechRecognition

3rd module: datetime

4th module: pip install wikipedia

5th module: pip install win32com.client (or) smtplib

Speak()
Datatime()
Wishme()
takeCommandCMD()
takeCommandMIC()

refer : https://www.youtube.com/watch?v=XWQXMncxg4k&t=111s

```
import speech_recognition as sr
import pyttsx3
import datetime
import webbrowser
import wikipedia
import pyjokes
import pyautogui
from plyer import notification
from bs4 import BeautifulSoup
import requests

engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)


def input_query():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print('recognition is on....')
        recognizer.pause_threshold = 0.7
        voice = recognizer.listen(source)
        try:
            query = recognizer.recognize_google(voice).lower()
            print('this is the query that was made....', query)
            return query
        except Exception as ex:
```

```
            print('An exception occurred', ex)


def report_time():
    current_time = datetime.datetime.now().strftime('%I:%M %p')
    return current_time


def speak_va(transcribed_query):
    engine.say(transcribed_query)
    engine.runAndWait()

def make_request(url):
  response = requests.get(url)
  return response.text

def activate_va():
    user_query = input_query()
    print('user query ....', user_query)
    if 'time' in user_query:
        current_time = report_time()
        print(f"the current time is {current_time}")
        speak_va(f"the current time is {current_time}")
    elif 'open website' in user_query:
        speak_va(
            "Please type the name of the website that you want to open (specify the full url) \n")
        website_name = input()
        print(website_name)
        webbrowser.get(
            'C:/Program Files (x86)/Google/Chrome/Application/chrome.exe
%s').open(website_name)
        speak_va(f"{website_name} opened.")
    elif 'wikipedia' in user_query:
        speak_va("Searching on Wikipedia")
        user_query = user_query.replace('wikipedia', ' ')
        result = wikipedia.summary(user_query, sentences=4)
        print(result)
        speak_va(result)
    elif 'joke' in user_query:
        random_joke = pyjokes.get_joke()
        print(random_joke)
        speak_va(random_joke)
    elif 'screenshot' in user_query:
        image = pyautogui.screenshot()
        image.save('screenshot.png')
        speak_va('Screenshot taken.')
```

```
    elif 'search' in user_query:
        speak_va("What do you want me to search for (please type) ? ")
        search_term = input()
        search_url = f"https://www.google.com/search?q={search_term}"
        webbrowser.get(
            'C:/Program Files (x86)/Google/Chrome/Application/chrome.exe %s').open(search_url)
        speak_va(f"here are the results for the search term: {search_term}")
    elif 'covid stats' in user_query:
        html_data = make_request('https://www.worldometers.info/coronavirus/')
        # print(html_data)
        soup = BeautifulSoup(html_data, 'html.parser')
        total_global_row = soup.find_all('tr', {'class': 'total_row'})[-1]
        total_cases = total_global_row.find_all('td')[2].get_text()
        new_cases = total_global_row.find_all('td')[3].get_text()
        total_recovered = total_global_row.find_all('td')[6].get_text()
        print('total cases : ', total_cases)
        print('new cases', new_cases[1:])
        print('total recovered', total_recovered)
        notification_message = f" Total cases : {total_cases}\n New cases : {new_cases[1:]}\n Total
Recovered : {total_recovered}\n"
        notification.notify(
            title="COVID-19 Statistics",
            message=notification_message,
            timeout=5
        )
        speak_va("here are the stats for COVID-19")
while True:
    activate_va()
```

pip install tensorflow
pip install keras
pip install OpenCV
import cv2 → OpenCV
pip install matplotlib
pip install numpy
pip install docker (https://docker-py.readthedocs.io/en/stable/)
pip install Podman (https://pypi.org/project/podman/)
pip install treelib (https://treelib.readthedocs.io/en/latest/)
pip install **pyautogui (it works well to take screenshots and identify images, works on both win and mac, refer: Automating the boring stuff)**

To Store data in different formats

Refer: https://realpython.com/python-sqlite-sqlalchemy/

- **Flat files** for data storage

- **SQL** to improve access to persistent data
- **SQLite** for data storage
- **SQLAlchemy** to work with data as Python objects

Creating, Reading & Updating A Config File with Python

Subtitle: Avoid Hard-Coded Data While Making Your (And Others) Life Easier
When the subject of config, or configuration, files comes up there is often confusion on what they are, how to read them, and the benefits that come from their use. In this article, we will cover what config files are before introducing the Python module `configparser.py`. We'll then use some examples to help you to understand, create, read, and update config files of your own.

A Quick Introduction To Configuration Files

While primarily used by operating systems to customise an environment, config files may be used for a broad set of uses. The most common use cases for coders or developers is to configure initial settings for a programme without hard coding data into the programme itself. This allows you to change those settings without the need to enter the programme and edit the source code itself. So in short, those things that may change over time such as IP addresses, host-names, user names, passwords, database names, can all be structured into a config file and the programme directed to that file for the specific information required. For those who wish to provide public access to their code on a repository like GitHub, config files may be included in the `.gitignore` file to ensure they remain inaccessible to the public, thereby protecting sensitive information.

How Are They Structured?

I am not aware of any standards or conventions concerning the layout of config files. Several formats allow quite complex data structures to be easily stored for later recovery and use. The deciding factor on the format you use will come down to what your programming language most easily handles and your own personal preference. Given our interest in Python, I'll concentrate on one common format but really, the sky is the limit on what you can use. Common formats include INI, JSON, YAML and XML. Today we'll focus on the INI format.

Creating, Reading, And Updating A config.ini File

Back in the day, Microsoft Windows relied largely on initialisation files, or INI files, to configure operating system and applications settings. In Python, we have a module called `configparser.py` which allows us to easily create, read and update INI files. If you don't already have it installed, use the command

```
$ pip install configparser
```

Structure of an INI file

Each INI file you create will consist of sections within which data is stored using key-value pairs in much the same way as a **Python dictionary** datatype. The following is a handwritten sample of an INI file which we will create, then we will read from it, before finally updating it:
Planned Filename: configfile.ini

```
[postgresql]
host = localhost
user = finxter1
password = myfinxterpw
db = postgres

[user_info]
```

```
admin = David Yeoman
login = finxter_freelancer
password = freelancer_freedom
```

Note that this proposed INI file consists of two sections, `'postgresql'` and `'user_info'`. Under each section is the data keys (host, user, password etc.) and the data values (`localhost, finxter1, myfinxterpw` etc.).

So this `configfile.ini` might be useful if our code needs to create an SQL query to our Postgres database, or if we need to sign in to a particular site requiring admin privileges, login, and password data.

You can add as many sections as you wish for multiple use cases.

Creating a new config file

Open a new Python file in which we will write the code to create the INI file. The following code is largely self-explanatory. Note that in this code I'm assuming no file of the same name currently exists hence the `'w'` command for `'write'` in the second to last line of the code. Here are the steps

Import the `configparser` module

Create the `configparser` object and pass it to `'config'`

Create a section name.

Populate the section with key: value pairs (such as `'host' = 'local_host'` in our example)

Repeat items 3 and 4 for any subsequent sections.

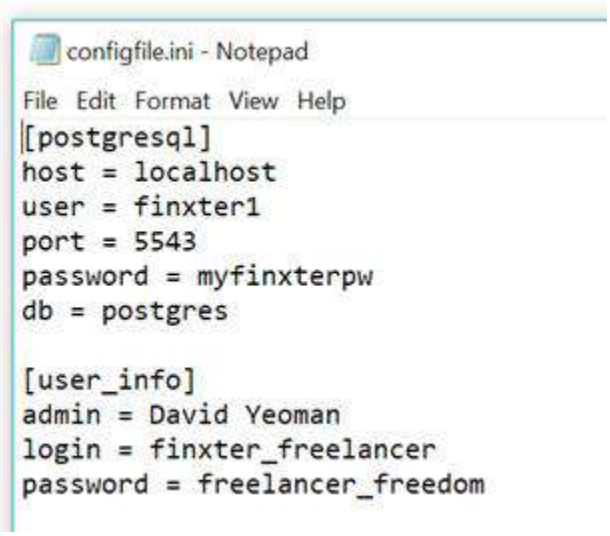
Finally, open a file of your chosen name at your chosen directory location using `'w'` to write

Write the file

So here is the code that follows the steps above.

```
import configparser
config = configparser.ConfigParser()
# Add the structure to the file we will create
config.add_section('postgresql')
config.set('postgresql', 'host', 'localhost')
config.set('postgresql', 'user', 'finxter1')
config.set('postgresql', 'port', '5543')
config.set('postgresql', 'password', 'myfinxterpw')
config.set('postgresql', 'db', 'postgres')
config.add_section('user_info')
config.set('user_info', 'admin', 'David Yeoman')
config.set('user_info', 'login', 'finxter_freelancer')
config.set('user_info', 'password', 'freelancer_freedom')
# Write the new structure to the new file
with open(r"C:\PythonTutorials\configfile.ini", 'w') as configfile:
config.write(configfile)
```

Execute the file and check the directory you entered in the code to find the new file. Here is a screenshot of the file I've created with the code above.



Congratulations. You've written your first INI file. Now, how do we use it?

Reading a config file

For the sake of this exercise, let us assume you wish to create an SQL query from some new source code you're writing. How would we read the config file and use the data? For those of you who aren't into SQL let's do a quick run-through of how you would structure this without a config file, then we'll show you how to structure it with one. It will be easy to see the difference. Here are the steps without a config file.

Import SQLAlchemy (my preferred tool for Postgresql queries)
You next need to set all the parameters that will form a URI to allow connection to the database
Create your URI and your connection engine and pass it to a variable
Connect to your database
Here is the code you'd use without using a config file.

```
import sqlalchemy
# set your parameters for the database connection URI
user = "finxter1"
password = "myfinxterpw"
host = "localhost"
port = 5543
dbase = "postgres"
# Create the URI and the engine with the URI in the following format
# postgresql://username:password@hostname:port/database
# therefore we pass the URI the parameters established above
conn = sqlalchemy.create_engine('postgresql://user:password@host:port/dbase',
encoding='utf8')
# Connect to the database
connection = conn.raw_connection()
```

So when we execute this code we're now connected to our database, but there's an obvious problem. We have hard-coded our username and password into the source code for any and all to see. Not clever.

Now let's use this same code with some additions to allow us to use the `configfile.ini` that we created above. We have two extra steps, `import configparser` and then read the `configfile.ini`. Here are the steps

Import SQLAlchemy
Import configparser
Read the configfile.ini
Read the parameters from the configfile.ini that will form a URI to connect to the database
Create your URI and your connection engine and pass it to a variable
Connect to your database

Here is the code you'd use when connecting using your INI file.

```
import sqlalchemy
import configparser
#Read config.ini file
config_obj = configparser.ConfigParser()
config_obj.read("C:\PythonTutorials\configfile.ini")
dbparam = config_obj["postgresql"]
useinfo = config_obj["user_info"]
# set your parameters for the database connection URI using the keys from the configfile.ini
user = dbparam["user"]
password = dbparam["password"]
host = dbparam["host"]
port = int(dbparam["port"])
dbase = dbparam["db"]
# Create the URI and the engine with the URI in the following format
# postgresql://username:password@hostname:port/database
# so we pass the URI the parameters established above
conn = sqlalchemy.create_engine('postgresql://user:password@host:port/dbase',
encoding='utf8')
# Connect to the database
connection = conn.raw_connection()
```

As you can see we needed to use `configparser` to create an object, then read the configfile.ini that we created earlier. Having read the configfile.ini we passed each section of the file to separate variables, in this case, the `'postgresql'` parameters to `dbparam` and the `'user_info'` parameters to useinfo.

```
#Read config.ini file
config_obj = configparser.ConfigParser()
config_obj.read("C:\PythonTutorials\configfile.ini")
dbparam = config_obj["postgresql"]
useinfo = config_obj["user_info"]
```

So instead of hard coding the values into the parameters for the database connection as we did previously, we can now call each value we need from the configfile.ini by using the keys and assigning them to variables.

```
# set your parameters for the database connection URI using the keys from the configfile.ini
user = dbparam["user"]
```

```
password = dbparam["password"]
host = dbparam["host"]
port = int(dbparam["port"])
dbase = dbparam["db"]
```

To prove that the data has been appropriately transferred, let's use a print command to check.

```
import sqlalchemy
import configparser
#Read config.ini file
config_obj = configparser.ConfigParser()
config_obj.read("C:\PythonTutorials\configfile.ini")
dbparam = config_obj["postgresql"]
useinfo = config_obj["user_info"]
# set your parameters for the database connection URI using the keys from the configfile.ini
user = dbparam["user"]
password = dbparam["password"]
host = dbparam["host"]
port = int(dbparam["port"])
dbase = dbparam["db"]
print('User variable = ', user, '\n')
print('Password variable = ', password, '\n')
print('Host variable = ', host, '\n')
print('Port variable = ', port, '\n')
print('Database variable = ', dbase, '\n')
# Result
User variable = finxter1
Password variable = myfinxterpw
Host variable = localhost
Port variable = 5543
Database variable = postgres
Process finished with exit code 0
```

So admittedly a bit more code in the second example, but look at the benefits you get.

You don't need to hard code sensitive information into a file that others can explore

No matter how many times you use the parameters throughout your code, if you need to change their values you simply edit the config file

If you wish to post your code on a repository like GitHub you can place the config file in the `.gitignore` file to be sure others cannot access your secret information, then you add a readme file to instruct others how to create a new config file with their own information

Updating your config file

This is simplicity itself. Here are two options.

The first is to open the `configfile.ini` in notepad and manually make the changes. Let's change the `'user'` details from `finxter1` to `coding_finxter8765`.

Now run the code again.

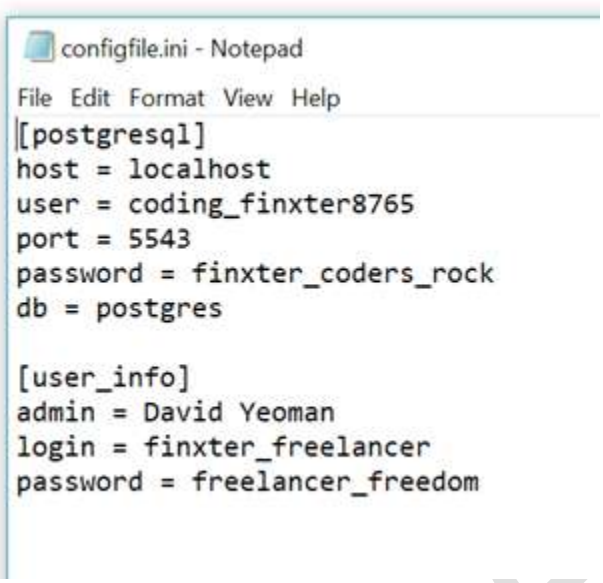
```
import sqlalchemy
import configparser
#Read config.ini file
config_obj = configparser.ConfigParser()
config_obj.read("C:\PythonTutorials\configfile.ini")
dbparam = config_obj["postgresql"]
useinfo = config_obj["user_info"]
# set your parameters for the database connection URI using the keys from the configfile.ini
user = dbparam["user"]
password = dbparam["password"]
host = dbparam["host"]
port = int(dbparam["port"])
dbase = dbparam["db"]
print('User variable = ', user, '\n')
print('Password variable = ', password, '\n')
print('Host variable = ', host, '\n')
print('Port variable = ', port, '\n')
print('Database variable = ', dbase, '\n')
# Result
User variable = coding_finxter8765
Password variable = myfinxterpw
Host variable = localhost
Port variable = 5543
Database variable = postgres
```

The second option is to use `configparser` from within Python. Let's change the PostgreSQL password from `myfinxterpw` to `finxter_coders_rock`. Here's the code.

```
import configparser
# Read config.ini file
edit = configparser.ConfigParser()
```

```
edit.read("C:/PythonTutorials/configfile.ini")
#Get the postgresql section
postgresql = edit["postgresql"]
#Update the password
postgresql["password"] = "finxter_coders_rock"
#Write changes back to file
with open('C:/PythonTutorials/configfile.ini', 'w') as configfile:
edit.write(configfile)
```

Here's the result.



To Summarise

Today we discussed configuration or `'config'` files. What they are, why they're used and some common formats used to structure them.

We introduced the `configparser` module used in Python and wrote a script to create a new INI file called `configfile.ini`.

We then illustrated the use (and perils) of hard-coded values when creating an SQL connection, before amending the SQL connection script to allow us to access the necessary values from the configfile.ini which we had created.

Finally, we demonstrated how easy it is to amend an INI file by manually amending some data with a word processor before we returned to the `configparser` module to make further amendments.

Remember – I have used an example here of creating a database connection, but you can use a config file for any data that you wish to remain secure, or for data that is often used throughout some source code and which may change from time to time. Any changes may be made once by editing the config file rather than sorting through hundreds of lines of code and manually editing all occurrences.

I hope this article has been of use to you. Thanks for reading.

Create web apps easily with streamlit

Pip install streamlit

Import streamlit as st

Please refer: https://awesome-streamlit.readthedocs.io/_/downloads/en/latest/pdf/

streamlit contact form

https://formsubmit.co/