====PYTHON-SCRIPTING-FOR-AUTOMATION===

print ("Hello World")

print("Welcome To Python Scripting")

--python executes from top to bottom.

Python is an interpreted language, which means the source code of a Python program is converted into bytecode that is then executed by the Python virtual machine. ... Python code is fast to develop: As the code is not needed to be compiled and built, Python code can be readily changed and executed.

Interpreter translates just one statement of the program at a time into machine code. Compiler scans the entire program and translates the whole of it into machine code at once. An interpreter takes very less time to analyze the source code. ... A compiler takes a lot of time to analyze the source code

--Special Characters

**Write special characters only inside of quotes----> ", ""
-\n -----> New Line
print("Hello World \nWelcome to Python")

Hello World
Welcome to Python

\b -----> back space

print("Hello World \bWelcome to Python")
Hello WorldWelcome to Python

\t -------> tab

\ --------> Escape char

For this kind of line

print('pyhtno's class')

We will get below error

```
  File "/Users/ASUTOSH/COURSES/python-scripting-automation/hello-world.py", line 7
    print('pyhtno's class')
                 ^
SyntaxError: invalid syntax
```

-We need to escape the character

print('pyhtno\'s class')

pyhtno's class

\a --------> To Create Alert

-VARIABLES

A variable is nothing but a reserved memory location to store values. In other words a variable in a program gives data to the computer to work on.

x = 10

print(f"The value of x is", x)

x = 10

print(f"The value of x is", x)

```
y = 20

z = "20"

print(type(y))

print(type(z))
```

--
The value of x is 10
<class 'int'>
<class 'str'>

Rules to define a variable names?
1.It contains letters, numbers and underscore. 2.It should not a keyword.
3.Can’t contain spaces.
4.It should not start with a number 5.Case-sensitive

--Datatypes

Every value in Python has a data type.
Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes. There are various data types in Python.
Basic Data Types are:
1. Numbers(int,floatandcomplex)
2. Strings
3. Boolean
Note: type(variable_name) will give the type of data stored in variable

**If we want to print the memeory location of the variable, we can get by below--

```
x = 10
```

```
print(id(x))
```

```
4384754608
```

--

```
#if we want to declare variables in a single line.
x = 5; y = 5.6; z = 4+6j
#If we want to print all the values in a single print.
print(x,y,z)
```

```
5 5.6 (4+6j)
```

```
print(x, type(x))
print(y, type(y))
print(z, type(z))
```

```
5 <class 'int'>
5.6 <class 'float'>
(4+6j) <class 'complex'>
```

--

```
myname = "Asutosh"
print(myname)
print(type(myname))
```

```
Asutosh
<str>
```

To comment all the lines at a time mention ['" lines "']

```
'''
x = 10

print(id(x)) #This will give us the memory location of x
```

```
#if we want to declare variables in a single line.
x = 5; y = 5.6; z = 4+6j
#If we want to print all the values in a single print.
print(x,y,z)
print(x, type(x))
print(y, type(y))
print(z, type(z))

myname = "Asutosh"
print(myname)
print(type(myname))
'''
```

---

boolean

```
myvalue = True
print(type(myvalue))
```

```
<class 'bool'>
```

```
x = 51
y = str(x)
z = bool(x)

print(x,type(x))
print(y,type(y))
print(z,type(z))
```

```
<class 'bool'>
51 <class 'int'>
51 <class 'str'>
True <class 'bool'>
```

-Any data type can be converted to boolean.

- Any data type can be converted to string.

- String data type can not be converted to integer.

```
x = None
print(bool(x))
```

```
False
```

-Print with multiple variables and strings

```
x = 3
y = 5.7
lang_name = "Python Scripting" #Either we can print like this or like below

print(x,y,lang_name)

print(f"The value of x is {x}")

print(f"The value of y is {y}")

print(f"The value of language name is {lang_name}")
```

--
```
3 5.7 Python Scripting
The value of x is 3
The value of y is 5.7
The value of language name is Python Scripting
```

--
#Or we can print like this

```
print(f"The values are {lang_name} \n {y} \n {x}")

The values are Python Scripting
 5.7
 3

 --
 #We can also print it like this

new_value = print(f"The values are {lang_name} \n {y} \n {x}")

print(new_value)

The values are Python Scripting
 5.7
 3
 ------------------------------------------
 -----Input and Output Syntax
 ---------------------------
'''
x = int(input("Enter the value of x" ))

y = int(input("Enter the value of y"))
'''
#This will only take the int value or the float value as mentioned.

#This will take any value we give and python will decide what type it is
x = eval(input("Enter the value of x" ))

y = eval(input("Enter the value of y")) <=========This will take any value and decide the type.

print(f"The value of x is {x} and the type of x is {type(x)}")

print(f"The value of x is {y} and the type of y is {type(y)}")
```

```
print(f"The addition of {x} and {y} is {x + y}")
```

--
Enter the value of x 25.5
Enter the value of y 25.5
The value of x is 25.5 and the type of x is <class 'float'>
The value of x is 25.5 and the type of y is <class 'float'>
The addition of 25.5 and 25.5 is 51.0

--
Enter the value of x "23"
Enter the value of y "25"
The value of x is 23 and the type of x is <class 'str'>
The value of x is 25 and the type of y is <class 'str'>
The addition of 23 and 25 is 2325

-Basic Operations on Strings

**A string is a sequence of characters.

**A character is simply a symbol. For example, the English language has 26 characters.

**Computers do not deal with characters, they deal with numbers (binary). Even though you may see characters on your screen, internally it is stored and manipulated as a combination of 0's and 1's.

**This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encoding used.

**In Python, string is a sequence of Unicode character. Unicode was introduced to include every character in all languages and bring uniformity in encoding

---
-Create String

---We can use double or single or triple quotes.

```
my_name = "Asutosh"
my_new_name = 'Python Developer'
my_info = """I have started my career as system admin and them moved to system
engineering"""

print(f"my name is: {my_name}\n my new name is: {my_new_name}\n my info is: {my_info}")
```

--
```
my name is: Asutosh
 my new name is: Python Developer
 my info is: I have started my career as system admin and them moved to system engineering
```

---If I will write like below

```
 my_name = "Asutosh"
my_new_name = 'Python Developer'
my_info = """I have started my career
as system admin and
them moved to system engineering"""

print(f"my name is: {my_name}\n my new name is: {my_new_name}\n my info is: {my_info}")
```

--The output will be like that only.

```
my name is: Asutosh
 my new name is: Python Developer
 my info is: I have started my career
as system admin and
them moved to system engineering
```

We can also print the string like this--
my_str = ""

my_new_str = " "

print(bool(my_str))
print(bool(my_new_str)) <=======space is a character

my_fav_scripting = "Pyhthon scripting"

print(my_fav_scripting)

--
False
True
Pyhthon scripting

-How to access the specific characters in a given string

my_fav_scripting = "Pyhthon scripting"
012345678910111213 14
----------->

It can also work with negative values
my_fav_scripting = "Pyhthon"
-7-6-5-4-3-2-1

-If we ant the last charactor of the string is my_fav_scripting[-1]

-If I would like to print some characters my_fav_scripting[0:5] #Starting from 0 you will get 5 characters.

-print my_fav_scripting[0:] #This means 0 and last

---
my_fav_scripting = "Pyhthon scripting"

print(my_fav_scripting[-1])
print(my_fav_scripting[0:])
print(my_fav_scripting[0:5])

g
Pyhthon scripting
Pyhth

--This is called slicing of a string

 -change or delete string

 -Strings are immutable. This means that elements of a string cannot be changed once it has been assigned. We can simply reassign different strings to the same name)

  We can delete and we can assign new string

     del my_fav_scripting

my_fav_scripting = "python new scripting"

print(my_fav_scripting)

-Calculate length of the scripting

my_fav_scripting = "python new scripting"

print(my_fav_scripting)

print(len(my_fav_scripting))

--
python new scripting
21

--How to add two strings
my_fav_scripting = "python new scripting"

my_fav_laptop = "macbook"

print(my_fav_scripting + my_fav_laptop)

--
python new scriptingmacbook

---Case(Lower,Upper,tiltel etc) conversion operations
---
my_fav_scripting = "python new scripting"

print(my_fav_scripting.upper())
print(my_fav_scripting.lower())
print(my_fav_scripting.title())

PYTHON NEW SCRIPTING
python new scripting
Python New Scripting

--
*******To listout all the operations are available on a given string.

print(dir(my_fav_scripting))

['__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']

---Boolean result operations

    -Lets check some operations.

my_string = "Python Scripting"

print(my_string.startswith("p"))
print(my_string.startswith("P"))

--
False
True

**We can use these operations on the conditional loops.

print(my_string.startswith("Python"))

print(my_string.endswith("ting"))

True
True

--
print(my_string.isupper())

```
print(my_string.istitle())

False
True

--
print(my_string.isnumeric())

False
------------------------------------------------------
----Join,center and zfill(zero fill)
-----------------------------------
 #JOIN
x = "Python"
print("*".join(x))
print("\n".join(x))

--
P*y*t*h*o*n
P
y
t
h
o
n
--
print("-".join(x))

P-y-t-h-o-n
---
#CENTER

#CENTER
my_string = "python"
mynewstring = "python scripting"
```

```
myoldstring = "system admin"

print(my_string.center(20))

     python

---
print(f"{my_string.center(30)}\n{mynewstring.center(30)}\n{myoldstring.center(30)}")

        python
       python scripting
        system admin

-----
#zfill

#zerofill(zfill)
my_string = "python"
print(my_string.zfill(20)) #This operation is called as padding.

--
00000000000000python

   #It will fill the left side of the string with 20 zeros.

-----------------------------------------------------------------
--Strip,split operations
------------------------
#Strip
y = "python"
print(y)

#If unwantedly I have given multiple spaces at the left side or right side of the string

x = "Python"
```

```
print(x.strip()) #So it will strip all the spaces  towards left or towards rightand give us the result.


x = "Python"
print(x.strip()) #So it will strip all the spaces  towards left or towards rightand give us the result.

#We can also remove one character either starting side or ending side.

print(x.strip('P'))
print(x.strip('n'))
```

--

```
python
Python

ython
Pytho
```

We can not remove the characters in between the string

--

I want to remove the right side of the python

```
x = "Python is python"

print(x.rstrip("python"))
```

--

```
Python is
```

--

```
#SPLIT

x = "python is easy"
```

```
print(x.split()) #It will convert the string into a list.

--
['python', 'is', 'easy']

--
print(x.split("is"))

--
['python ', ' easy']
-----------------------------------------------------------------------
--Count,index and find operations on strings
----------------------------------------
#count
x = "Python is a very easy scripting language"

print(x.count("s"))
print(x.count("is"))

--
3
1
---
#index

#index
x = "Python is a very easy scripting language"
print(x.index("s"))#This will print the first s.

print(x.index("s", 9))#It will search the indexes after 8, from 9 and give us the next index.

--
8
19
```

#find
x = "Python is a very easy scripting language"

print(x.find("easy"))

print(x.find("l"))

print(x.find("r", 5))

--
17
32
14

----Practice: Display given string at left/right/center of a line in title format
---

***To know the columns in a linux operating system command line

#tput cols
156

-To take the default column size in all the operating systems automatically, we need to import os module.

#Practice: Display given string at left/right/center of a line in title format
import os
t_w = os.get_terminal_size().columns
my_string = input("Enter the string:")
print(my_string.center(t_w).title())
print(my_string.ljust(t_w).title())
print(my_string.rjust(t_w).title())

#os.terminal_size().columns <=====This will get the terminal column size on any platform.

--
Enter the string: python scripting
python scripting
python scripting

------Data Structures Of Python

--Introduction to data structures and types of data structures
---
* Data structures are also variables.

*Data Structures are used to store a collection of data.

*There are four built-in data structures.
They are:
List --> []
Tuple-->()
Dictionary --> {} with key value pair
Set-->{}

--List
---
my_value = ["asutosh", "ankita", "sarang", "prajakta", 3, 10, 90]

Lists are mutable. We can change value through index.

my_value = ["asutosh", "ankita", "sarang", "prajakta", 3, 10, 90]
empthy_list = []
print(my_value[1], type(my_value))
print(bool(my_value))
print(bool(empthy_list)) #boolean of empty list is false.

```
print(my_value[-1]) #it will print the last value

print(my_value[3][1]) #SO it will show us the index[1] of the string/value[3] "prajakta" index 1 is
r
print(my_value[:]) #This will give me the entire list

print(my_value[1:3]) #it will print the index 1 and 3.
print(my_value[1:]) #it will print from 1 to last
#If we want to assign a value to any index
my_value[4] = 100
print(my_value)
----
ankita <class 'list'>
True
False
90
r
['asutosh', 'ankita', 'sarang', 'prajakta', 3, 10, 90]
['ankita', 'sarang']
['ankita', 'sarang', 'prajakta', 3, 10, 90]
['asutosh', 'ankita', 'sarang', 'prajakta', 100, 10, 90]

----------
--Operations for List
--------------------
These are the operations we can do in list.

>>> my_value = ['asutosh', 'ankita', 'sarang', 'prajakta', 100, 10, 90]
>>> print(dir(my_value))
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__',
'__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
```

'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

--
print(my_value.index(90)) # It will tell the index of value 90
print(my_value.count(90)) # It will count the number of times
#print(my_value.clear()) # It will clear the list
#print(my_value)
print(my_value.append(200)) #It will add the value given into the list
my_value.insert(1,5)
print(my_value)

--
6
1
None
['asutosh', 5, 'ankita', 'sarang', 'prajakta', 3, 10, 90, 200]

---Tuples

Tuple and strings are immutible.
my_value_empty = ()

my_value = ("asutosh", "sarang", "shailesh",5, 8, 10, ["1,2,5"])

print(my_value, type(my_value))
print(bool(my_value_empty))
print(my_value[6][0]) #we can get the index of the list and the index of that list values
#my_value[2] = "saswat" # we can not change tupple value.
#print(my_value)

('asutosh', 'sarang', 'shailesh', 5, 8, 10, ['1,2,5']) <class 'tuple'>
False
1,2,5

```
Traceback (most recent call last):
  File "/Users/ASUTOSH/COURSES/python-scripting-automation/tuple.py", line 8, in
<module>
    my_value[2] = "saswat"
TypeError: 'tuple' object does not support item assignment
```

We do not have more list of operations

```
my_value = ('asutosh', 'ankita', 'sarang', 'prajakta', 100, 10, 90)
>>> print(dir(my_value))
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__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']
--------------------------------
--Dictionaries
--------------
my_dict = {}

my_dict_new = {"fruit": "apple", "name": "asutosh", "count": 3, "number": 5}
print(type(my_dict))
print(my_dict_new)
#print(my_dict_new[1]) #we can not print dictinary with index number
print(my_dict_new["name"])
----
<class 'dict'>
{'fruit': 'apple', 'name': 'asutosh', 'count': 3, 'number': 5}
asutosh

my_dict_new["new"] = 6 #This is the syntax to add a new value to the dictionary. If key is
already there, then new value will be given otherwise new key:value will be added.
print(my_dict_new)
```

--
{'fruit': 'apple', 'name': 'asutosh', 'count': 3, 'number': 5, 'new': 6}

--Operations on dictionary
---
my_dict_new = {"fruit": "apple", "name": "asutosh", "count": 3, "number": 5}
>>> print((dir(my_dict_new))
...)
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__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']

--
print(my_dict_new.values())
print(my_dict_new.keys())

--
dict_values(['apple', 'asutosh', 3, 5])
dict_keys(['fruit', 'name', 'count', 'number'])

--set
---
my_set_empty = set({}) # If we want to define an empty set.
my_set = {1,19,20,30,45,76,89,90} #set will order the values, if we have duplicate entries then it will print uniq value.
print(my_set)
print(type(my_set_empty))

--
{1, 76, 45, 19, 20, 89, 90, 30}
<class 'set'>

---
list = [3,4,5,6,7,8,9,0,4,5,6,7]
print(set(list)) #we can convert the list into set, and will give us uniq data.

--
{0, 3, 4, 5, 6, 7, 8, 9}

Different types of operations on set

>>> set = {1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6}
>>> print(set)
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

>>> print(dir(set))
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset', 'pop', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union', 'update']

-----Operators of python

----Introduction to operators of python.

Operators:
-Operators are the pillars for any language.
-The operator can be defined as a symbol which is responsible for a particular operation between two operands.
-Example ‘+’ is an operators , to perform addition operation on operands. (x and y are operands and the + is the operator)

-Operands may be values or variables or combination of values and variables.
-Python provides a variety of operators described as follows.

Arithmetic Operators

Assignment Operators

Comparison Operators

Identity Operators

Membership Operators

Logical Operators

Bitwise Operators

--
Arithmetic Operators
Assignment Operators
Takes values as inputs, performs its operation on input values and gives values as outputs.
Takes values as inputs, performs its operation on input values and gives output as either True or False.
Takes True or False as inputs, performs operation on this inputs and give output as either True or False
Comparison Operators
Identity Operators
Membership Operators

Logical Operators
Bitwise Operators
Takes values as inputs, performs operations on its binary representation and gives output as a value.

--Arithmatic and assignment operators

Arithmetic Operators:
Addition +
Subtraction -
Multiplication *
Division /
Modulo %
Floor division //
Exponential **

```
x = 8
y = 9

z = x**2
n = x%2
m = x/2
print(z)
print(n)
print(round(m))
```

--
```
64
0
4
```

Assignment Operators:

oprator	Example	Same as
=	a = b	a = b
+=	a += b	a = a+b
-=	a -= b	a = a-b

```
 *=              a *= b              a = a*b
 /=              a /= b              a = a/b
 %=               a %= b               a = a%b

'''x = 8
y = 9

z = x**2
n = x%2
m = x/2
print(z)
print(n)
print(round(m))'''

x = eval(input("Enter first number"))
y = eval(input("Enter second number"))
z = x + y
print(f"The result of {x} and {y} is {z}")

-----------------------------------------
--Comparision Operators
----------------------
x == y

x > y

x < y

x != y

x <= y

x >= y

---
```

```
x = 9
y = 89
if x > y:
   print("True")
else:
   print("false")
```

****Python is comparing the variables by converting it into a ASCII code. To check that

```
x = 9
y = 89
if x > y:
   print("True")
else:
   print("false")

print(ord("x")) #We can get the ASCII code of the characters.
print(ord("y"))
```

--

```
false
120
121
```

--

We can also get the charactor

```
x = 9
y = 89
if x > y:
   print("True")
else:
   print("false")

print(ord("x")) #We can get the ASCII code of the characters.
```

```
print(ord("y"))
#we can also find our charactor from the ASCII code.
print(ord("+"))
print(chr(43))
print(chr(45))
--
false
120
121
43
+
-
```

--Identity and Membership operators

Identity operators:

Identity operators are used to find the type of: class/type/object . They are two types of Identity operators:

is

is not

```
---
x = 67.0

y = 87

print(type(x))
print(type(y))

z = type(x) is type(y)
print(z)
----
<class 'float'>
<class 'int'>
False
```

Membership operators:
Membership operators are used to validate the membership of a value. They are two types of Membership operators:
in
notin

```
a = [1,2 ,3 ,4 ,5 ,6 ,7 ,8 ]
b = 6 in a
print(b)
```

--
True

--
```
a = [1,2 ,3 ,4 ,5 ,6 ,7 ,8 ]
b = 6 not in a
print(b)
```

--
False

```
valid_java = ["2.8","1.8"]
host_java = "1.8"
if host_java in valid_java:
   print("The host deployed with valid java version")
else:
   print("The host deployed with invalid java version")
```

The host deployed with valid java version

---Logical Operators

Logical operators are useful to combine multiple conditions. They are three type of logical operators in python.
They are:
**and
**or
**not

For AND operator – It returns TRUE if both the operands (right side and left side) are true

For OR operator- It returns TRUE if either of the operand (right side or left side) is true

For NOT operator- returns TRUE if operand is false

```
x = 8
y = 10
if (x <= y and x == y):
   print("False")
elif (x < y or y > x):
   print("True")
else:
   print("oops")
```

True

```
x = 8
y = 8
if (x <= y and x == y):
   print("False")
elif (x < y or y > x):
   print("True")
else:
   print("oops")
```

False

```
x = 8
y = 8
if (x < y and x == y):
   print("False")
elif (x < y or y > x):
   print("True")
else:
   print("oops")
```
---
oops

--Conditional Statements

----Introduction to conditional statements

- if is calld simple conditional statement. used to control the execution of set of lines or block of code or one line.

-How to validate the conditions.
comparison operators
identity
membership
logical operators

```
usr_string = input("Enter the string: ")
usr_cnf = input("Do you want to convert it to the lower case? say yes or no: ")
if usr_cnf == "yes":
   print(usr_string.lower())
```

Enter the string: PYTHON
Do you want to convert it to the lower case? say yes or no: yes

python

```
my_even_number = [0,2,4,6,8,10]

usr_input = eval(input("Enter the number: "))

if usr_input in my_even_number:
   print("The given number is even")
else:
   print("The number is not in the list")
```

```
Enter the number: 25
The number is not in the list
```

---if---else and if---elif conditions

****PRACTICE

```
#Read a number between 1-10 and display it in words

#Read a number between 1-10 and display it in words
'''usr_number = eval(input("Enter the number between 1 to 10"))
if usr_number == 1:
   print("ONE")
elif usr_number == 2:
   print("TWO")
elif usr_number == 3:
   print("THREE")
elif usr_number == 4:
   print("FOUR")
elif usr_number == 5:
   print("FIVE")
elif usr_number == 6:
   print("SIX")
elif usr_number == 7:
```

```
    print("SEVEN")
elif usr_number == 8:
    print("EIGHT")
elif usr_number == 9:
    print("NINE")
elif usr_number == 10:
    print("TEN")
else:
    print("Please enter the number between 1 to 10")
 '''
 ----------------------------------------------------------------
 #instead of writing so many lines we can use a dictionary.
user_number = eval(input("Enter the number between 1 to 10: "))
num = {1:"ONE", 2:"TWO", 3:"THREE", 4:"FOUR", 5:"FIVE", 6:"SIX", 7:"SEVEN", 8:"EIGHT",
9:"NINE", 10:"TEN"}

if user_number in [1,2,3,4,5,6,7,8,9,10]:
     print("The number is valid", num.get(user_number))

else:
     print("Enter the number between 1 to 10")

---------
Enter the number between 1 to 10: 8
The number is valid EIGHT
-----------------------------------------------------------------
-----Working with Python Modules
------------------------------
--Introduction to Python modules
------------------------------
 What is a module ?
A module is a file containing Python definitions and statements. That means, module containing
python functions, classes and variables.

What is the use of module ?
```

Reusability

Note: If script name is mymodule.py then module name is mymodule

Types of Python Modules:
* Default Modules
* Third Party Modules

Import either default or third party modules before using them.

*Different ways to import module:
Using import math Using from:
from math import *
from math import pi,pow

---
mymodule.py

#We can resuse the python scripts as modules in other python scripts.
my_value = 12345678909876543 21

#Now we will import this value to first-script.py

---
first-script.py
---
import mymodule #Now this script will be imported here as a amodule without .py extension.

print(mymodule.my_value)

12345678909876543 21

--How to know what are the default modules are there with python.

#help("modules")

--Lets I want to work with math module, I want to see all the operations available with math module.

>>> import math
>>> dir(math)
['__doc__', '__file__', '__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', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

e.g
---
import math
print(math.pow(9,2))
print(math.pi)

81.0
3.141592653589793

--If we want to see the documentation then we can do
import math
help(math)

If you want to install a third party module, then you can use pip.

sh-3.2# pip3 install xlwt
Collecting xlwt
  Downloading
https://files.pythonhosted.org/packages/44/48/def306413b25c3d01753603b1a222a011b8621aed27cd7f89cbc27e6b0f4/xlwt-1.3.0-py2.py3-none-any.whl (99kB)
    |████████████████████████████████| 102kB 191kB/s

Installing collected packages: xlwt
Successfully installed xlwt-1.3.0
WARNING: You are using pip version 19.2.3, however version 21.2.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.

----If you want to work with aws, then there is a modules called "boto3"

sh-3.2# pip3 install boto3
Collecting boto3
  Downloading
https://files.pythonhosted.org/packages/69/c9/486ca332f093e4f93545c646eb06ffe549f367bc273ecd77d0e7ab7d7b78/boto3-1.18.31-py3-none-any.whl (131kB)
    |████████████████████████████████| 133kB 282kB/s
Collecting botocore<1.22.0,>=1.21.31 (from boto3)
  Downloading
https://files.pythonhosted.org/packages/60/9c/58569911efe366f7ee1ee294ebeceba06d0e73780038749afd27595ed03f/botocore-1.21.31-py3-none-any.whl (7.8MB)
    |████████████████████████████████| 7.8MB 910kB/s
Collecting jmespath<1.0.0,>=0.7.1 (from boto3)
  Downloading
https://files.pythonhosted.org/packages/07/cb/5f001272b6faeb23c1c9e0acc04d48eaaf5c862c17709d20e3469c6e0139/jmespath-0.10.0-py2.py3-none-any.whl
Collecting s3transfer<0.6.0,>=0.5.0 (from boto3)
  Downloading
https://files.pythonhosted.org/packages/ab/84/fc3717a7b7f0f6bb08af59312717 1f08e3e0087c197922da09c01bfe7c3a/s3transfer-0.5.0-py3-none-any.whl (79kB)
    |████████████████████████████████| 81kB 22.0MB/s
Requirement already satisfied: urllib3<1.27,>=1.25.4 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (from
botocore<1.22.0,>=1.21.31->boto3) (1.26.4)
Collecting python-dateutil<3.0.0,>=2.1 (from botocore<1.22.0,>=1.21.31->boto3)
  Downloading
https://files.pythonhosted.org/packages/36/7a/87837f39d0296e723bb9b62bbb257d0355c7f6128853c78955f57342a56d/python_dateutil-2.8.2-py2.py3-none-any.whl (247kB)
    |████████████████████████████████| 256kB 968kB/s

```
Collecting six>=1.5 (from python-dateutil<3.0.0,>=2.1->botocore<1.22.0,>=1.21.31->boto3)
  Downloading
https://files.pythonhosted.org/packages/d9/5a/e7c31adbe875f2abbb91bd84cf2dc52d792b5a015
06781dbcf25c91daf11/six-1.16.0-py2.py3-none-any.whl
Installing collected packages: six, python-dateutil, jmespath, botocore, s3transfer, boto3
Successfully installed boto3-1.18.31 botocore-1.21.31 jmespath-0.10.0 python-dateutil-2.8.2
s3transfer-0.5.0 six-1.16.0
WARNING: You are using pip version 19.2.3, however version 21.2.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
----------
If you want to work with remote servers, then we can install one third party module called
"paramico"

sh-3.2# pip3 install paramiko
Collecting paramiko
  Downloading
https://files.pythonhosted.org/packages/95/19/124e9287b43e6ff3ebb9cdea3e5e8e88475a873c0
5ccdf8b7e20d2c4201e/paramiko-2.7.2-py2.py3-none-any.whl (206kB)
    |████████████████████████████████| 215kB 240kB/s
Collecting bcrypt>=3.1.3 (from paramiko)
  Downloading
https://files.pythonhosted.org/packages/bf/6a/0afb1e04aebd4c3ceae630a87a55fbfbbd94dea4ea
f01e53d36743c85f02/bcrypt-3.2.0-cp36-abi3-macosx_10_9_x86_64.whl
Collecting pynacl>=1.0.1 (from paramiko)
  Downloading
https://files.pythonhosted.org/packages/d4/68/a84e1cc99e4b08f857f148ba5ff6653afb954e121e
8362c7a6242d8755ef/PyNaCl-1.4.0-cp35-abi3-macosx_10_10_x86_64.whl (380kB)
    |████████████████████████████████| 389kB 2.6MB/s
Collecting cryptography>=2.5 (from paramiko)
  Downloading
https://files.pythonhosted.org/packages/00/e4/da1509e64a92e32ec8df97f5c4372e7f0e56b5b0b
ad299da61a9632b900c/cryptography-3.4.8-cp36-abi3-macosx_10_10_x86_64.whl (2.0MB)
    |████████████████████████████████| 2.0MB 1.3MB/s
Collecting cffi>=1.1 (from bcrypt>=3.1.3->paramiko)
```

```
  Downloading
https://files.pythonhosted.org/packages/ca/e1/015e2ae23230d9de8597e9ad8c0b81d5ac181f08f
2e6e75774b7f5301677/cffi-1.14.6-cp38-cp38-macosx_10_9_x86_64.whl (176kB)
     |████████████████████████████████| 184kB 1.5MB/s
Requirement already satisfied: six>=1.4.1 in
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages (from
bcrypt>=3.1.3->paramiko) (1.16.0)
Collecting pycparser (from cffi>=1.1->bcrypt>=3.1.3->paramiko)
  Downloading
https://files.pythonhosted.org/packages/ae/e7/d9c3a176ca4b02024debf82342dab36efadfc5776f
9c8db077e8f6e71821/pycparser-2.20-py2.py3-none-any.whl (112kB)
     |████████████████████████████████| 112kB 40kB/s
Installing collected packages: pycparser, cffi, bcrypt, pynacl, cryptography, paramiko
Successfully installed bcrypt-3.2.0 cffi-1.14.6 cryptography-3.4.8 paramiko-2.7.2 pycparser-2.20
pynacl-1.4.0
WARNING: You are using pip version 19.2.3, however version 21.2.4 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
-----------------------------------------
-We can also import modules and use them as an alias name.

import platform as pt
print(pt.system())

---
Darwin
---------------
We can import multiple modules in single script

import platform as pt
import os
import sys
import subprocess
------------------------------------------------------------
---Platform Modules
-------------------
```

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

```
>>> dir(platform)
['DEV_NULL', '_UNIXCONFDIR', '_WIN32_CLIENT_RELEASES',
'_WIN32_SERVER_RELEASES', '__builtins__', '__cached__', '__copyright__', '__doc__',
'__file__', '__loader__', '__name__', '__package__', '__spec__', '__version__',
'_comparable_version', '_component_re', '_default_architecture', '_dist_try_harder',
'_follow_symlinks', '_ironpython26_sys_version_parser', '_ironpython_sys_version_parser',
'_java_getprop', '_libc_search', '_linux_distribution', '_lsb_release_version', '_mac_ver_xml',
'_node', '_norm_version', '_parse_release_file', '_platform', '_platform_cache',
'_pypy_sys_version_parser', '_release_filename', '_release_version', '_supported_dists',
'_sys_version', '_sys_version_cache', '_sys_version_parser', '_syscmd_file', '_syscmd_uname',
'_syscmd_ver', '_uname_cache', '_ver_output', '_ver_stages', 'architecture', 'collections', 'dist',
'java_ver', 'libc_ver', 'linux_distribution', 'mac_ver', 'machine', 'node', 'os', 'platform', 'popen',
'processor', 'python_branch', 'python_build', 'python_compiler', 'python_implementation',
'python_revision', 'python_version', 'python_version_tuple', 're', 'release', 'subprocess', 'sys',
'system', 'system_alias', 'uname', 'uname_result', 'version', 'warnings', 'win32_ver']
```

To get help with the platform module

```
#help(platform)
```

There is a built-in function to list all the function names (or variable names) in a module and that is dir() function.
Then use dir() function as:
print(dir(platform))
How to get help of a platform module ?
From script: print(help(platform))
From Python command line: help(platform)

```
>>> import platform

>>> print(f"This is {platform.system()} OS")
```

This is Darwin OS

>>> print(f"The python version is {platform.python_version()}")
The python version is 3.8.1

>>> platform.node()
'ASUTOSH-MACBOOK-PRO.local'
>>> platform.processor()
'i386'
>>> platform.uname()
uname_result(system='Darwin', node='ASUTOSH-MACBOOK-PRO.local', release='19.6.0', version='Darwin Kernel Version 19.6.0: Thu May 6 00:48:39 PDT 2021; root:xnu-6153.141.33~1/RELEASE_X86_64', machine='x86_64', processor='i386')

--getpass module

(getpass() and getuser())

**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.

****In linux we can give the python path in vi, and we can run the python script without python.

sh-3.2# chmod +x read-password.py
sh-3.2# ./read-password.py
Enter the db password:
The database passowrd is 123

sh-3.2# cat read-password.py
#! /Library/Frameworks/Python.framework/Versions/3.8/bin/python3

'''db_pass = input("Enter password")'''
#With this password would be visible while typing
import getpass
db_pass = getpass.getpass(prompt= "Enter the db password: ") #if I want to prompt for password
print(f"The database passowrd is {db_pass}")

--getuser()
---
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.

#! /Library/Frameworks/Python.framework/Versions/3.8/bin/python3
'''db_pass = input("Enter password")'''
#With this password would be visible while typing
import getpass
db_pass = getpass.getpass(prompt= "Enter the db password: ") #if I want to prompt for password
print(f"The database passowrd is {db_pass}")

print(getpass.getuser())

Enter the db password:
The database passowrd is 123
ASUTOSH

-------SYS Module

--Introduction to sys module

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

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)

---
import sys
print(sys.api_version)
print(sys.version)

-----
import sys
print(sys.api_version)

print(sys.version_info)
#print(sys.modules)
#print(sys.path)#environement variable for python.
sys.exit() #It will stop running the python script. It will not execute anything after this.
print("Hello")

--sys.argv of sys module(VEry very important)
---
sys.argv returns a list of command line arguments passed to a Python script.

for example---

python hello.py hi how are you <=======It will not affect the result of the script.

import sys
print(sys.argv)

**When we will execute the script

/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/sys-module.py python scripting automation

OUTPUT---
['/Users/ASUTOSH/COURSES/python-scripting-automation/sys-module.py', 'python', 'scripting', 'automation']

It will give the list of the "command line arguments" given

If we will put the index as 0

print(sys.argv[0])
It will give us only the script name.
/Users/ASUTOSH/COURSES/python-scripting-automation/sys-module.py

---
For example we want to enter a string and the action through command line argument.

```
import sys
#We are not going to use input commadn here.We will use command line arguments
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 validate option like (lower/upper/title)")
```

/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/command-line-arguments.py
python upper

PYTHON

#We can also add one error message here, if the length of the arguments are less than 3

```
import sys
#We are not going to use input commadn here.We will use command line arguments
#We can also add one error message here, if the length of the arguments are less than 3
if len(sys.argv) != 3:
    print("Usage:")
    print(f"{sys.argv[0]} <string-name> <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 validate option like (lower/upper/title)")
```

---
python command-line-arguments.py python
Usage:
command-line-arguments.py <string-name> <lower|upper|title>

--
python command-line-arguments.py python lower
python

--------OS Module

--Introduction to OS Module and Basic operations

This module is used to work/interact with operating system to automate many more tasks like creating directory, removing directory, identifying current directory and many more.

Operations of os module Four parts of os
---
os
os.path
os.system()
os.walk()

Simple os operations
---
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()

---If want to get the details of the os functions.

```
>>> import os
>>> dir(os)
```

['CLD_CONTINUED', 'CLD_DUMPED', 'CLD_EXITED', 'CLD_TRAPPED', 'DirEntry', 'EX_CANTCREAT', 'EX_CONFIG', 'EX_DATAERR', 'EX_IOERR', 'EX_NOHOST', 'EX_NOINPUT', 'EX_NOPERM', 'EX_NOUSER', 'EX_OK', 'EX_OSERR', 'EX_OSFILE', 'EX_PROTOCOL', 'EX_SOFTWARE', 'EX_TEMPFAIL', 'EX_UNAVAILABLE', 'EX_USAGE', 'F_LOCK', 'F_OK', 'F_TEST', 'F_TLOCK', 'F_ULOCK', 'MutableMapping', 'NGROUPS_MAX', 'O_ACCMODE', 'O_APPEND', 'O_ASYNC', 'O_CLOEXEC', 'O_CREAT', 'O_DIRECTORY', 'O_DSYNC', 'O_EXCL', 'O_EXLOCK', 'O_NDELAY', 'O_NOCTTY', 'O_NOFOLLOW', 'O_NONBLOCK', 'O_RDONLY', 'O_RDWR', 'O_SHLOCK', 'O_SYNC', 'O_TRUNC', 'O_WRONLY', 'PRIO_PGRP', 'PRIO_PROCESS', 'PRIO_USER', 'P_ALL', 'P_NOWAIT', 'P_NOWAITO', 'P_PGID', 'P_PID', 'P_WAIT', 'PathLike', 'RTLD_GLOBAL', 'RTLD_LAZY', 'RTLD_LOCAL', 'RTLD_NODELETE', 'RT

***In case of windows always give the path in below format

c:\\Users\Automation (As the \n, \t are there python will give error with the single \ in case of windows.)

/Users/ASUTOSH/COURSES/python-scripting-automation <==========In linux it is ok

------
```
import os
print(os.sep)
#print(os.getcwd())
#os.chdir("/Users/ASUTOSH/COURSES/python-scripting-automation")
#print(os.getcwd())
#print(os.listdir())
#print(os.listdir("/Users/ASUTOSH/COURSES/python-scripting-automation"))
#os.mkdir("Asutosh")
print(os.environ)
print(os.getpid())
print(os.getuid())
```

--os.path module(This is a sub module of os module)

---
os.path is a sub module of os (os.path module is used to work on paths

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)

---
path = "/Users/ASUTOSH/COURSES/python-scripting-automation"
print(os.path.basename(path))
print(os.path.dirname(path))

---
python-scripting-automation
/Users/ASUTOSH/COURSES

-os.system() function from os module
---
os.system() from os module

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

--
print(os.system("pwd"))
print(os.system("ls -lh"))

```
---
/Users/ASUTOSH/COURSES/python-scripting-automation
0
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/os-module.py
/Users/ASUTOSH/COURSES/python-scripting-automation
0
total 216
drwxr-xr-x  2 ASUTOSH  admin    64B Aug 31 02:25 Asutosh
-rw-r--r--  1 ASUTOSH  admin   369B Aug 25 02:47 Practice.py
drwxr-xr-x  4 ASUTOSH  admin   128B Aug 30 13:49 __pycache__
-rw-r--r--  1 ASUTOSH  admin   209B Aug 26 05:15 arithmatic-assignment-operator.py
-rw-r--r--  1 ASUTOSH  admin   257B Aug 19 04:52 boolean-result.py
-rw-r--r--  1 ASUTOSH  admin   605B Aug 30 15:51 command-line-arguments.py
-rw-r--r--  1 ASUTOSH  admin   243B Aug 26 05:28 comparison.py
-rw-r--r--  1 ASUTOSH  admin   378B Aug 28 23:46 conditional-statement.py
-rw-r--r--  1 ASUTOSH  admin   426B Aug 19 23:46 count-index-find-operations.py

---------------
#We will right a asimple script.
cmd = "datee" #here we have given wrong command. If the command is wrong, then it will give a
error value.
rt = os.system(cmd)
print(rt)
if rt == 0:
   print("The command completed succdessfully")
else:
   print("command was failed")

---
sh: datee: command not found
32512
command was failed
```

#We will right a asimple script.
cmd = "date" #here we have given wrong command. If the command is wrong, then it will give a error value.
rt = os.system(cmd)
print(rt)
if rt == 0:
    print("The command completed succdessfully")
else:
    print("command was failed")

Tue Aug 31 03:01:02 IST 2021
0
The command completed succdessfully

--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)

#(write a single python script to clear terminal of any Operating system)
import os
import platform

if platform.system() == "Windows":
    os.system("cls")
else:
    os.system("clear")

It will clear the screen.

--OS.walk(path)

Used to generate directory and file names in a directory tree by walking.

--For example

```
import os
path = "/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh"

print(list(os.walk(path)))

print("======================")
#with for loop we can separate the each tupple it is creating for each path
for i in list(os.walk(path)):
    print(i)
```

```
[('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh', ['Ankita', 'jimmy'],
['test1', 'test2']), ('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/Ankita',
[], []), ('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/jimmy', [], [])]
```

*****The output is a whole list, inside that a tuple, inside that 3 lists

1st list - display the path

2nd list - display the list of directories under that path

3rd list - display the list of files under that path

```
[('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh', ['Ankita', 'jimmy'],
['test1', 'test2']), ('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/Ankita',
[], []), ('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/jimmy', [], [])]
======================
('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh', ['Ankita', 'jimmy'],
['test1', 'test2'])
```

('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/Ankita', [], [])
('/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/jimmy', [], [])

-----So with for loop we can differentiate the tuples.

If I will differentiate the output in 3 parts, root, directory lists and the file lists inside the root path.

```
import os
path = "/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh"

'''#print(list(os.walk(path)))
print("========================")
#with for loop we can separate the each tupple it is creating for each path
for i in list(os.walk(path)):
    print(i)'''
for r,d,f in os.walk(path):
    print(r)    <==============It will only display the root paths under that particular path..
```

```
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/Ankita
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/jimmy
```

```
import os
path = "/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh"

'''#print(list(os.walk(path)))
print("========================")
#with for loop we can separate the each tupple it is creating for each path
for i in list(os.walk(path)):
    print(i)'''
for r,d,f in os.walk(path):
    print(r,d)
```

```
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh ['Ankita', 'jimmy']
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/Ankita []
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/jimmy []
---------
import os
path = "/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh"

'''#print(list(os.walk(path)))
print("========================")
#with for loop we can separate the each tupple it is creating for each path
for i in list(os.walk(path)):
    print(i)'''
for r,d,f in os.walk(path):
    print(r,d,f)
----------    ----------
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh ['Ankita', 'jimmy'] ['test1',
'test2']
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/Ankita [] []
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/jimmy [] []

-----------------------
import os
path = "/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh"

'''#print(list(os.walk(path)))
print("========================")
#with for loop we can separate the each tupple it is creating for each path
for i in list(os.walk(path)):
    print(i)'''
for r,d,f in os.walk(path):
    if len(f) !=0:  #It will display the file list where the length  of file lists is not equal to zero
        print(r)
        #for i in f:
        print(f)
-----------------
```

/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh
['test1', 'test2']

I want to display the files also separately

```
import os
path = "/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh"

'''#print(list(os.walk(path)))
print("========================")
#with for loop we can separate the each tupple it is creating for each path
for i in list(os.walk(path)):
    print(i)'''
for r,d,f in os.walk(path):
    if len(f) !=0:  #It will display the file list where the length  of file lists is not equal to zero
        print(r)
        for i in f:
          print(i) #It will display the files separately without the list.
```

/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh
test1
test2

-Now for each and evry file I want to get complete path

```
import os
path = "/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh"

'''#print(list(os.walk(path)))
print("========================")
#with for loop we can separate the each tupple it is creating for each path
for i in list(os.walk(path)):
    print(i)'''
```

```
for r,d,f in os.walk(path):
   if len(f) !=0:  #It will display the file list where the length  of file lists is not equal to zero
      print(r)
      for i in f:
        #print(i) #It will display the files separately without the list.
         print(os.path.join(r,i)) #this will join the output and give us the full path of the file
```

```
--------------------
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/test1
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/test2
```

```
-----------------------------------------------------------------------------------------
--How to do a system wide search for a file.
-----------------------------------------
```

```
#How to do a system wide search for a file with os.walk
import os
required_file = input("Enter the file name to search: ")

for r,d,f in os.walk("/"):
   for each_file in f:
      if each_file == required_file:
         print(os.path.join(r,each_file)) #It will join the outputs and we will get the full path of the
file.
```

```
-----------------------------------------------------------------------------------------
----Read a path and check if given path is a file or a directory
-------------------------------------------------------------
#Read a path and check if given path is a file or a directory
import os
path = input("Enter the path: ")
#before chaking of the file or directory we need to check if the path is existing or not.
if os.path.exists(path):
```

```
    print(f"The given path {path} is valid ")
    if os.path.isfile(path):
        print(f"The given path: {path} is a file")
    else:
        print(f"The given path: {path} is a directory")
else:
    print(f"The given path {path} is not a valid path")
```

```
Enter the path: /Users/ASUTOSH/COURSES/python-scripting-automation/xyz
The given path /Users/ASUTOSH/COURSES/python-scripting-automation/xyz is not a valid
path
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/os.walk-practice2.py
Enter the path: /Users/ASUTOSH/COURSES/python-scripting-automation
The given path /Users/ASUTOSH/COURSES/python-scripting-automation is valid
The given path: /Users/ASUTOSH/COURSES/python-scripting-automation is a directory
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/os.walk-practice2.py
Enter the path: /Users/ASUTOSH/COURSES/python-scripting-automation/hello-world.py
The given path /Users/ASUTOSH/COURSES/python-scripting-automation/hello-world.py is
valid
The given path: /Users/ASUTOSH/COURSES/python-scripting-automation/hello-world.py is a
file
```

--Read a directory path and identify all files and directories(Intro to for loop)

If we will run as below

```
#Read a directory path and identify all files and directories
import os
path = input("Enter your directory path: ")
print(os.listdir(path))
```

Enter your directory path: /Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh
['test1', 'Ankita', 'test2', 'jimmy'] <=====

The output will be a list of directory and files.

No we can use a for loop to find out the results

```
#Read a directory path and identify all files and directories
import os
import sys
path = input("Enter your directory path: ")
if os.path.exists(path):
  print("The given path is valid")
  df_l = os.listdir(path)
else:
   print("Please provide a avalid path")
   sys.exit()
#so if we need to identify the values are directory or file, I need to separate.
#But first we need to get the complete path of the files/directories
#Now we can do it through for loop
list_dir = os.listdir(path)
for i in list_dir:
   l_f_d = os.path.join(path,i) #this will join and print all the dir/files under that path
   if os.path.isfile(l_f_d):
      print(f"{l_f_d} is a file")
   else:
      print(f"{l_f_d} is a directory")
```

Enter your directory path: /Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh
The given path is valid
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/test1 is a file
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/Ankita is a directory
/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/test2 is a file

/Users/ASUTOSH/COURSES/python-scripting-automation/Asutosh/jimmy is a directory

---
--Loops | Working with for loop

Execute a block of code many times)

Why Loops:

All programming languages need a way to execute block of code many times, this is possible with loops.
Python has two types of loops:

for loop
while loop

for loop:

The for loop in Python is used to iterate over a sequence (list, tuple, string) or other iterable objects.

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

---
for loop

For example--

```
#print the index numbers of the given string
my_string = input("Enter the desired string: ")
j = 0
for i in my_string:
```

```
  print(f"{i} =======> index {j}")  <========== i will itereate with string again and again. It is called iteration.
  j = j+1

-------------
Enter the desired string: ASUTOSH
A =======> index 0
S =======> index 1
U =======> index 2
T =======> index 3
O =======> index 4
S =======> index 5
H =======> index 6
---------------

my_list = [3,4,9,5,8,6,20,25]
for i in my_list:
    rem = i % 2
    if rem == 0:
        print(f"The given number {i} is even number")
    else:
        print(f"The given number {i} is odd number")

-------
The given number 3 is odd number
The given number 4 is even number
The given number 9 is odd number
The given number 5 is odd number
The given number 8 is even number
The given number 6 is even number
The given number 20 is even number
The given number 25 is odd number
-----------------------------------------------
--Simple practice with for loop
------------------------------
```

Read a string and print chars and their index values

```
#print the index numbers of the given string
my_string = input("Enter the desired string: ")
j = 0
for i in my_string:
   print(f"{i} =======> index {j}")  <========== i will itereate with string again and again. It is called iteration.
   j = j+1
```

```
Enter the desired string: ASUTOSH
A =======> index 0
S =======> index 1
U =======> index 2
T =======> index 3
O =======> index 4
S =======> index 5
H =======> index 6
```

-- Find all files in a directory with required extension .py/.sh/.log/.txt etc...

```
#Find all files in a directory with required extension .py/.sh/.log/.txt etc...
import os
req_path = input("Enter the required path: ")
if os.path.isfile(req_path):
    print(f"The given path {req_path} is a file. Please enter the path of the directory")
else:
    req_ext = input("Enter the required file extention.(.py/.sh/.txt): ")
    list_dir = os.listdir(req_path)
    for i in list_dir:
        if i.endswith(req_ext): #As the file names are string we are adding the function endswith
            print(i)
```

```
Enter the required path: /Users/ASUTOSH/COURSES/python-scripting-automation
Enter the required file extention.(.py/.sh/.txt): .py
```

multiple-variables.py
practice.os.walk.py
tuple.py
os-module.py
identity-membership.py
list.py
count-index-find-operations.py
os-walk-practice3.py

If I want to display the files in a list

```
#Find all files in a directory with required extension .py/.sh/.log/.txt etc...
import os
req_path = input("Enter the required path: ")
if os.path.isfile(req_path):
    print(f"The given path {req_path} is a file. Please enter the path of the directory")
else:
    req_ext = input("Enter the required file extention.(.py/.sh/.txt): ")
    list_dir = os.listdir(req_path)
    req_list = []
    for i in list_dir:
        if i.endswith(req_ext): #As the file names are string we are adding the function endswith
            #print(i)
            req_list.append(i) #If we want to diosplay the result in a list.
    print(req_list)
```

```
Enter the required path: /Users/ASUTOSH/COURSES/python-scripting-automation
Enter the required file extention.(.py/.sh/.txt): .txt
['test1.txt', 'test.txt']
```

--Complete Range Function

(Python3 range() is a an object)

range() function:

The range() function generates the integer numbers between the given start integer to the stop integer, which is generally used to iterate over with for Loop.

Python3 range() accepts an integer and returns a range object, which is nothing but a sequence of integers.

Syntax:

range (start, stop[, step])

range() takes three arguments.

Out of the three 2 arguments are optional. I.e., Start and Step are the optional arguments.

Default value of start is 0 and step is 1

syntax

range(start,stop,step)

bydefault 3 values it is taking. ------> If no start is mentioned default value is 0.

```
>>> print(list(range(1,20)))
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] <====Here no step mentioned so by default step /difference is 1

```
>>> print(list(range(1,20,3)))   <======here the step is given as 3/difference is 3 between the numbers
```

```
#print all the even numbers between 0 to 100.
print(list(range(2,100,2))) #start is 2, stop is 100(upto 99) , step/difference is 2

#print all the odd numbers between 0 to 100
print(list(range(1,100,2)))  #start is 1, stop is 100(upto 99) , step/difference is 2

for i in range(0,20):
    print(i)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98]
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99]
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
-----------------------------------------------------
-- For loop for string, list, tuple and dictionaries
-----------------------------------------------------
'''my_string = "Working with for loop"
print(my_string)

#for i in my_string: #If we want to print the characters one by one
    #print(i)
print("\n".join(my_string))# this is also we can use

#-------------#
#list
my_list = [1,2,3,4,5,6,7]
for i in my_list:
```

```
    print(i)'''
#tuple
'''my_list = [(1,2), (4,5), (6,7)]
#for i in my_list:
    #print (i)
#we can also separate the values with for loop
for i,j in my_list:    #we are assigning first value to i and second value to j
    print(i)
    print(j)'''
my_dict = {"a": 1, "b": 2, "c": 3}
#for i in my_dict:
    #print(i) #If we will take one variable then we will get only keys
for i,j in my_dict.items(): #to get the key and value from the dictonary we need to write this way

   #print(i)
   #print(j)
   print(i,j) #we can print this way or that way
   #It is very important in case of dictionary. We need to take key and value.

------------------------------------------------------------
--Introduction to while loop
----------------------------
 While loop:
The while loop in Python is used to iterate over a block of code as long as the test expression (condition) is true.
We generally use this loop when we don't know beforehand, the number of times to iterate.

----
'''#lets say I want to print a line 10 times.Then we can use for loop.
for i in range(10):
  print("WELCOME")

#lets say I want to execute a message infinite time

while True:
```

```
   print("-----------------------WELCOME--------------------")
#It will print infinitie number of times
'''
-----------
#lets say I want to monitor the filesystem for lifetime
'''import time
while True:
   time.sleep = 10
   print("Monitoring Filesystem")'''
---------------
#lets say I want to stop the while loop at a point

value = 4
while value <= 6789:
   value = value + 459
   print(value)
----------------
count = 0
while count <= 5:
   print("WELCOME")
   count = count+1
-----------------
```

-- Loop Control statements

Control statements:
break
continue
pass

The Python Break and Continue Statements are two important statements used to alter the flow of a program in any programming language.

Loops are used to execute certain block of statements for n number of times until the test condition is false.

There will be some situations where, we have terminate the loop without executing all the statements. In these situations we can use Python Break statement and Python Continue statements.

break statement:
The break statement terminates the loop containing it.

continue Statement:
The continue statement is used to skip the rest of the code inside a loop.

pass:
We use pass statement to write empty loops. Pass is also used for empty control statement, function and classes.

```
for i in [1,2,3,4,5,6,7]:
    print(i)
    break
 #Now if we will add break here, this will stop the loop/not script
print("WELCOME")
#Now because of break this will stop after the first iteration.
```
---
ol.py
1
WELCOME

```
#Continue
for i in range(1,20):
    if i == 7:
        continue #This will skip all the line of codes after "continue" when i = 7.
    print(i)
```

1
2
3
4
5
6
8
9
10
11
12
13
14
15
16
17
18
19

```
#pass
#pass
for i in range(1,20): #here nothing is written so will get error to avoid this we can use pass
  pass
```


--Introduction to datetime module

Python built in or default module and used to work with dates and times

datetime module:
It is having different classes to manipulate date and time.

Simply import datetime module
and run dir(datetime) to know all operations/classes/functions/variables

```
>>> import datetime

>>> dir(datetime)
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
'__name__', '__package__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys', 'time',
'timedelta', 'timezone', 'tzinfo']

---------
import datetime
print(datetime.datetime.now())
print(datetime.datetime.today())

#if we want to display the time with timezone, we have to create a object.
import pytz
ist = pytz.timezone('Asia/kolkata')
print(datetime.datetime.now(ist))

print(datetime.datetime.now().year)

#If you want to display just the year-month-daye, you need to format the result as a string

print(datetime.datetime.now().strftime("%Y-%m-%d"))
#To findout all the parameters we can give to strftime(strftime.org)

print(datetime.datetime.now().strftime("%c"))

print(datetime.datetime.fromtimestamp(1555555))
-------------
2021-09-03 06:08:50.559889
2021-09-03 06:08:50.559944
2021-09-03 06:08:50.605316+05:30
2021
2021-09-03
Fri Sep  3 06:08:50 2021
```

1970-01-19 05:35:55

#If I want to use the submodule datetime directly
from datetime import datetime
print(datetime.now())
print(datetime.max)
print(datetime.min)

2021-09-03 06:13:22.797928
2021-09-03 06:13:22.797977
2021-09-03 06:13:22.836054+05:30
2021
2021-09-03
Fri Sep 3 06:13:22 2021
1970-01-19 05:35:55
2021-09-03 06:13:22.836156
9999-12-31 23:59:59.999999
0001-01-01 00:00:00

--Find all the files which are older than x days from a given path
---
```
import os
import sys
import datetime
age = 600
req_path = input("Enter the path: ")
if not os.path.exists(req_path):
    print("Enter valid path")
    sys.exit(1)
if os.path.isfile(req_path):
    print("please enter the directory path")
    sys.exit(2)

list_dir = os.listdir(req_path)
#As we are working with files we need to filter the complete path
```

```
today = datetime.datetime.now()
for i in list_dir:
    each_file_path = os.path.join(req_path,i)
    if os.path.isfile(each_file_path): #here we will get all the paths like dir/file, I want to skip the
dir paths
        file_cre_date = datetime.datetime.fromtimestamp(os.path.getctime(each_file_path))
        #print(each_file_path,datetime.datetime.fromtimestamp(os.path.getctime(each_file_path)))
        #print(dir(today - file_cre_date))
        diff_days = (today - file_cre_date).days
        if diff_days > age:
            print(each_file_path,diff_days)
            #os.remove(each_file_path)
#This os.path.getctime(each_file_path will give us the ctime in seconds
#this datetime.datetime.fromtimestamp(os.path.getctime(each_file_path)) convert the seconds
in date and time
#I do not bother about the hours I want to print in days output.
#print(each_file_path,(today - file_cre_date).days)
#If we want to find out which function we can use, we can run
#print(dir(today - file_cre_date))
#Now if we want to remove this thing, we need to concentrate on os module
-----------------------------------
/Users/ASUTOSH/Downloads/testfilenew.txt 608
/Users/ASUTOSH/Downloads/output.txt 608
/Users/ASUTOSH/Downloads/CHANGELOG-6.0.1EBF.txt 608
/Users/ASUTOSH/Downloads/Memory_details 608
/Users/ASUTOSH/Downloads/AccountTransactions04092019622085.txt 608
/Users/ASUTOSH/Downloads/Asutosh HR letter.pdf 608
/Users/ASUTOSH/Downloads/cldb.log.20191218.hc2n2.gz 608
/Users/ASUTOSH/Downloads/PPR000804042409.pdf 608
/Users/ASUTOSH/Downloads/maprcli_dump_volumenodes 608
/Users/ASUTOSH/Downloads/lonsl1103385.uk.net.intra-2019-11-27_12-59-07.tar 608
/Users/ASUTOSH/Downloads/Support-dump-collection.docx 608
/Users/ASUTOSH/Downloads/Attachment_1.html 608
/Users/ASUTOSH/Downloads/DC_EPS.vmp-epsmaster1.tar.gz 608
/Users/ASUTOSH/Downloads/zookeeper.outr-2790 608
```

/Users/ASUTOSH/Downloads/disktab.out 608
/Users/ASUTOSH/Downloads/destvolinfo 608
/Users/ASUTOSH/Downloads/command_output.txt 608
/Users/ASUTOSH/Downloads/gdb1.out 608
/Users/ASUTOSH/Downloads/Attachment_1 (3).eml 608

--Subprocess Module

- Used to execute Operating System commands

-SO in general while we are using os.system("df -h") command and to store the output in to a variable, we can not do that.

```
import os

x = os.system("df -h")
print(x)
```

```
Filesystem                                      Size   Used  Avail Capacity iused      ifree %iused  Mounted on
/dev/disk1s5                                    466Gi   10Gi  197Gi     6%  488378 4881964502    0%   /
devfs                                           191Ki  191Ki    0Bi   100%     662          0  100%   /dev
/dev/disk1s1                                    466Gi  255Gi  197Gi    57% 1211422 4881241458    0%   /System/Volumes/Data
/dev/disk1s4                                    466Gi  2.0Gi  197Gi     2%       2 4882452878    0%   /private/var/vm
map auto_home                                     0Bi    0Bi    0Bi   100%       0          0  100%   /System/Volumes/Data/home
/Users/ASUTOSH/Downloads/Visual Studio Code.app  466Gi  255Gi  197Gi    57% 1210576 4881242304    0%   /private/var/folders/s1/0xdvm8rs2k7bf5lok2ttj3r40000gn/T/AppTranslocation/A8E53CCD-E735-498A-B046-44C8DED1439F
0
```

**It will only show us either 0 or 1, if the command exceuted successfully then we iwll get "0" and if the command is not right we will get "1"

-This is why we are using subprocess module.

```
>>> import subprocess
>>> dir(subprocess)
['CalledProcessError', 'MAXFD', 'PIPE', 'Popen', 'STDOUT', '_PIPE_BUF', '__all__',
'__builtins__', '__doc__', '__file__', '__name__', '__package__', '_active',
'_args_from_interpreter_flags', '_cleanup', '_demo_posix', '_demo_windows',
'_eintr_retry_call', '_has_poll', 'call', 'check_call', 'check_output', 'errno', 'fcntl', 'gc',
'list2cmdline', 'mswindows', 'os', 'pickle', 'select', 'signal', 'sys', 'traceback', 'types']
```

---How to declare it.

```
import os
import subprocess

sp = subprocess.Popen(cmd,shell=True/False,stdout=subprocess.PIPe,stderr=subprocess.PIPE)
rc = sp.wait()
out,err = sp.communicate()
print(f"The output is {out}")
print(f"The error is {err}")
#===============#
#If I am taking the cmd as a string, then I can add the "shell = True", In tis case python is going to open unix like shell and run the command.
#cmd = "dir"
#cmd = "ls -lrt"
#If I am using cmd as a list, then we have to mention shell= False
#cmd = ["ls", "-lrt"]
--------------------------------------------------------------------------------
import subprocess
cmd = "ls"
sp = subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE)
rc = sp.wait()
```

```
out,err = sp.communicate()
print(f"The return code is: {rc}")
print(f"The output is: \n{out}")
print(f"The error is: \n{err}")

--------------
The return code is: 0
The output is:
b'Asutosh\nPractice.py\n__pycache__\narithmatic-assignment-operator.py\nboolean-result.py
\ncommand-line-arguments.py\ncomparison.py\ncomplete-range-function.py\nconditional-stat
ement.py\ncount-index-find-operations.py\ndatatype.py\ndate-time.py\ndelete_files_oldertha
n.py\ndictionary.py\nfind-all-files-for-loop.py\nfirst-script.py\nfor-loop-string-index.py\nfor.py
\nhello-world.py\nidentity-membership.py\ninput-output.py\njoin-center-zfill[zero-fill].py\nlis
t.py\nlogical-operators.py\nloop-control.py\nloops-for-data-string.py\nmultiple-variables.py\n
mymodule.py\nos-module.py\nos-walk-practice3.py\nos.walk-practice2.py\nos.walk.py\npracti
ce-conditionals.py\npractice-os-platform.py\npractice.os.walk.py\nread-password.py\nset.py\n
strings.py\nstrip-split-operations.py\nsubprocess-module.py\nsys-module.py\ntest.txt\ntest1.tx
t\ntuple.py\nvariable.py\nwhile-loop.py\n'
The error is:
b''
------------
import subprocess
cmd = "ll"
sp = subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE)
rc = sp.wait()
out,err = sp.communicate()
print(f"The return code is: {rc}")
print(f"The output is: \n{out}")
print(f"The error is: \n{err}")
-------------
The return code is: 127
The output is:
b''
The error is:
b'/bin/sh: ll: command not found\n'
```

---Actually we are getting the output as a byte code, to avoid that if we want to get the output as a string--

```
import subprocess
cmd = "ls -l"
sp =
subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_n
ewlines=True)
rc = sp.wait()
out,err = sp.communicate()
print(f"The return code is: {rc}")
print(f"The output is: \n{out}")
print(f"The error is: \n{err}")
```

The return code is: 0
The output is:
total 336
drwxr-xr-x 6 ASUTOSH admin   192 Sep 1 02:18 Asutosh
-rw-r--r-- 1 ASUTOSH admin   369 Aug 25 02:47 Practice.py
drwxr-xr-x 4 ASUTOSH admin   128 Aug 30 13:49 __pycache__
-rw-r--r-- 1 ASUTOSH admin   209 Aug 26 05:15 arithmatic-assignment-operator.py
-rw-r--r-- 1 ASUTOSH admin   257 Aug 19 04:52 boolean-result.py
-rw-r--r-- 1 ASUTOSH admin   605 Aug 30 15:51 command-line-arguments.py
-rw-r--r-- 1 ASUTOSH admin   243 Aug 26 05:28 comparison.py
-rw-r--r-- 1 ASUTOSH admin   297 Sep 3 04:39 complete-range-function.py
-rw-r--r-- 1 ASUTOSH admin   378 Aug 28 23:46 conditional-statement.py
-rw-r--r-- 1 ASUTOSH admin   426 Aug 19 23:46 count-index-find-operations.py
-rw-r--r-- 1 ASUTOSH admin   480 Aug 18 05:14 datatype.py
-rw-r--r-- 1 ASUTOSH admin   742 Sep 3 06:18 date-time.py
-rw-r--r-- 1 ASUTOSH admin 1420 Sep 3 07:18 delete_files_olderthan.py

The error is:

If I want to diplay the output in a list----

```
import subprocess
cmd = "ls"
sp =
subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_n
ewlines=True)
rc = sp.wait()
out,err = sp.communicate()
print(f"The return code is: {rc}")
print(f"The output is: \n{out.splitlines()}")
print(f"The error is: \n{err.splitlines()}")
```

The return code is: 0
The output is:
['Asutosh', 'Practice.py', '__pycache__', 'arithmatic-assignment-operator.py', 'boolean-result.py', 'command-line-arguments.py', 'comparison.py', 'complete-range-function.py', 'conditional-statement.py', 'count-index-find-operations.py', 'datatype.py', 'date-time.py', 'delete_files_olderthan.py', 'dictionary.py', 'find-all-files-for-loop.py', 'first-script.py', 'for-loop-string-index.py', 'for.py', 'hello-world.py', 'identity-membership.py', 'input-output.py', 'join-center-zfill[zero-fill].py', 'list.py', 'logical-operators.py', 'loop-control.py', 'loops-for-data-string.py', 'multiple-variables.py', 'mymodule.py', 'os-module.py', 'os-walk-practice3.py', 'os.walk-practice2.py', 'os.walk.py', 'practice-conditionals.py', 'practice-os-platform.py', 'practice.os.walk.py', 'read-password.py', 'set.py', 'strings.py', 'strip-split-operations.py', 'subprocess-module.py', 'sys-module.py', 'test.txt', 'test1.txt', 'tuple.py', 'variable.py', 'while-loop.py']
The err

If we ant to run the program with shell=False option, we can mention the cmd as a list.

```
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 return code is: {rc}")
print(f"The output is: \n{out.splitlines()}")
print(f"The error is: \n{err.splitlines()}")
#here python will not open a new shell
```

```
The return code is: 0
The output is:
['total 336', '-rw-r--r--  1 ASUTOSH  admin   129 Aug 17 23:12 hello-world.py', '-rw-r--r--  1 ASUTOSH  admin    88 Aug 18 04:15 variable.py', '-rw-r--r--  1 ASUTOSH  admin   425 Aug 18 05:13 multiple-variables.py', '-rw-r--r--  1 ASUTOSH  admin   480 Aug 18 05:14 datatype.py', '-rw-r--r--  1 ASUTOSH  admin   493 Aug 19 03:59 input-output.py', '-rw-r--r--  1 ASUTOSH  admin   913 Aug 19 04:42 strings.py', '-rw-r--r--  1 ASUTOSH  admin   257 Aug 19 04:52 boolean-result.py', '-rw-r--r--  1 ASUTOSH  admin   463 Aug 19 05:17 join-center-zfill[zero-fill].py', '-rw-r--r--  1 ASUTOSH  admin   502 Aug 19 23:33 strip-split-operations.py', '-rw-r--r--  1 ASUTOSH  admin   426 Aug 19 23:46 count-index-find-operations.py', '-rw-r--r--  1 ASUTOSH  admin   369 Aug 25 02:47 Practice.py', '-rw-r--r--  1 ASUTOSH  admin   895 Aug 25 03:22 list.py', '-rw-r--r--  1 ASUTOSH  admin   313 Aug 25 03:31 tuple.py', '-rw-r--r--  1 ASUTOSH  admin   405 Aug 25 03:57 dictionary.py', '-rw-r--r--  1 ASUTOSH  admin   335 Aug 25 04:04 set.py', '-rw-r--r--  1 ASUTOSH  admin   209 Aug 26 05:15 arithmatic-assignment-operator.py', '-rw-r--r--  1 ASUTOSH  admin   243 Aug 26 05:28 comparison.py', '-rw-r--r--  1 ASUTOSH  admin   331 Aug 26 05:49 identity-membership.py', '-rw-r--r--  1 ASUTOSH  admin   119 Aug 26 05:56 logical-operators.py', '-rw-r--r--  1 ASUTOSH  admin   378 Aug 28 23:46 conditional-statement.py', '-rw-r--r--  1 ASUTOSH  admin   970 Aug 29 16:55 practice-conditionals.py', '-rw-r--r--  1 ASUTOSH  admin   166 Aug 29 17:20 first-script.py', '-rw-r--r--  1 ASUTOSH  admin   248 Aug 29 17:33 mymodule.py', 'drwxr-xr-x  4 ASUTOSH  admin   128 Aug 30 13:49 __pycache__', '-rwxr-xr-x  1 ASUTOSH  admin   336 Aug 30 14:40 read-password.py', '-rw-r--r--  1 ASUTOSH  admin   262 Aug 30 15:04 sys-module.py', '-rw-r--r--  1 ASUTOSH  admin   605 Aug 30 15:51 command-line-arguments.py', '-rw-r--r--  1 ASUTOSH  admin   803 Aug 31 03:01 os-module.py', '-rw-r--r--  1 ASUTOSH  admin   186 Aug 31 03:16 practice-os-platform.py', 'drwxr-xr-x  6 ASUTOSH  admin   192 Sep  1 02:18 Asutosh', '-rw-r--r--  1 ASUTOSH  admin   608 Sep  1 03:09 os.walk.py', '-rw-r--r--  1 ASUTOSH  admin   259 Sep  1 03:12 practice.os.walk.py', '-rw-r--r--  1
```

ASUTOSH admin 477 Sep 1 03:42 os.walk-practice2.py', '-rw-r--r-- 1 ASUTOSH admin 726 Sep 1 04:39 os-walk-practice3.py', '-rw-r--r-- 1 ASUTOSH admin 165 Sep 3 02:45 for-loop-string-index.py', '-rw-r--r-- 1 ASUTOSH admin 204 Sep 3 03:26 for.py', '-rw-r--r-- 1 ASUTOSH admin 0 Sep 3 03:51 test.txt', '-rw-r--r-- 1 ASUTOSH admin 0 Sep 3 03:51 test1.txt', '-rw-r--r-- 1 ASUTOSH admin 630 Sep 3 04:08 find-all-files-for-loop.py', '-rw-r--r-- 1 ASUTOSH admin 297 Sep 3 04:39 complete-range-function.py', '-rw-r--r-- 1 ASUTOSH admin 892 Sep 3 05:03 loops-for-data-string.py', '-rw-r--r-- 1 ASUTOSH admin 670 Sep 3 05:26 while-loop.py', '-rw-r--r-- 1 ASUTOSH admin 474 Sep 3 05:52 loop-control.py', '-rw-r--r-- 1 ASUTOSH admin 742 Sep 3 06:18 date-time.py', '-rw-r--r-- 1 ASUTOSH admin 1420 Sep 3 07:18 delete_files_olderthan.py', '-rw-r--r-- 1 ASUTOSH admin 649 Sep 6 03:40 subprocess-module.py']
The error is:
[]

**SO when the env variables/outputs are involbed then we can use shell = True, else we can always use shell = False.

```
cmd = ["echo", "$HOME"]
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 return code is: {rc}")
print(f"The output is: \n{out.splitlines()}")
print(f"The error is: \n{err.splitlines()}")
```

If we will take shell = false in case of env variables, then we will get below output

The return code is: 0
The output is:
['$HOME']
The error is:

[]

We can get the list value of the command by below--

```
>>> cmd = "ls -lrt".split()
>>> cmd
['ls', '-lrt']
```

--Practice with subprocess module

```
#Find bash version of linux os
#Only the version
import subprocess
cmd = ["bash", "--version"] #list version of cmd
sp =
subprocess.Popen(cmd,shell=False,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_
newlines=True)
rc = sp.wait() #It will wait for sometime to execute the code
out,err = sp.communicate()

if rc == 0:
   print(f"The output is: {out.splitlines()}")#out.splitlines() convert the output in string
else:
   print(f"The command is not valid and the error is: {err.splitlines()}") #This will give us error if
the command is invalid
```

```
The output is: ['GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin19)', 'Copyright (C)
2007 Free Software Foundation, Inc.']
```

Now we are going to print the line one by one from the list

```
#Find bash version of linux os
#Only the version
import subprocess
cmd = ["bash", "--version"] #list version of cmd
sp =
subprocess.Popen(cmd,shell=False,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_
newlines=True)
rc = sp.wait() #It will wait for sometime to execute the code
out,err = sp.communicate()

if rc == 0:
    #print(f"The output is: {out.splitlines()}")#out.splitlines() convert the output in string
     for each_line in out.splitlines():
         print(each_line)
else:
    print(f"The command is not valid and the error is: {err.splitlines()}") #This will give us error if
the command is invalid
--------------
GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin19)
Copyright (C) 2007 Free Software Foundation, Inc.
---------------
Now I want to print the line with the version

#Find bash version of linux os
#Only the version
import subprocess
cmd = ["bash", "--version"] #list version of cmd
sp =
subprocess.Popen(cmd,shell=False,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_
newlines=True)
rc = sp.wait() #It will wait for sometime to execute the code
out,err = sp.communicate()

if rc == 0:
    #print(f"The output is: {out.splitlines()}")#out.splitlines() convert the output in string
```

```
    for each_line in out.splitlines():
        #print(each_line)
        if "version" in each_line and "release" in each_line:
            print(each_line.split())
else:
    print(f"The command is not valid and the error is: {err.splitlines()}") #This will give us error if
the command is invalid
```

```
GNU bash, version 3.2.57(1)-release (x86_64-apple-darwin19)
```

But I need only the version number(3.2.57(1))

```
#Find bash version of linux os
#Only the version
import subprocess
cmd = ["bash", "--version"] #list version of cmd
sp =
subprocess.Popen(cmd,shell=False,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_
newlines=True)
rc = sp.wait() #It will wait for sometime to execute the code
out,err = sp.communicate()

if rc == 0:
    #print(f"The output is: {out.splitlines()}")#out.splitlines() convert the output in string
    for each_line in out.splitlines():
        #print(each_line)
        if "version" in each_line and "release" in each_line:
            print(each_line.split()[3].split("(")[0]) #this will convert the output into string, index 3
value
else:       #split("(") This will split the value in two strings and we need the index 0
    print(f"The command is not valid and the error is: {err.splitlines()}") #This will give us error if
the command is invalid
```

['3.2.57', '1)-release']

(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/practice1-subprocess.py
3.2.57

--Platform independent script to find the java version

So here while getting the output, we are getting the output in err line as error

```
import subprocess
cmd = "java -version"
sp =
subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_n
ewlines=True)
rc = sp.wait()
out,err = sp.communicate()
print(out)
print(f"error {err}")
if rc == 0:
    print(f"The output is: {out}")
else:
    print(f"The given command is invalid and the error is {err.splitlines()}")
```

---
error java version "16.0.2" 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)

The output is:

****This is not the issue with the python, usually "java -version" is stored as error output

e.g -

[root@api ~]# java -version 1> /tmp/java.out <==========this command will capture the output into the file with command executed successfully with code 1. (1> means the command executed successfully)

openjdk version "1.8.0_272"
OpenJDK Runtime Environment (build 1.8.0_272-b10)
OpenJDK 64-Bit Server VM (build 25.272-b10, mixed mode)

If we will run the command to execute and capture the error into a file with option 2, then we can see the output.

java -version 2> /tmp/java.out 2 means to print the error in a file.

[root@api ~]# cat /tmp/java.out
openjdk version "1.8.0_272"
OpenJDK Runtime Environment (build 1.8.0_272-b10)
OpenJDK 64-Bit Server VM (build 25.272-b10, mixed mode)

If we want to nulify the output with 1 and 2

[root@api ~]# ls 1>/dev/null <usually with 1 means standard output, it will nulify the output with /dev/null

[root@api ~]# ls 2>/dev/null This is with error
anaconda-ks.cfg cats.txt disk.txt dogs.txt linux-shell-script python-scripts swapuse.sh terraform test.txt
---
import subprocess
cmd = "java -version"

```
sp =
subprocess.Popen(cmd,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,universal_n
ewlines=True)
rc = sp.wait()
out,err = sp.communicate()
if rc == 0:
   if bool(out) == True:
      print(out)
   if bool(out)==False and bool(err)==True:
       print(err.splitlines()[0].split()[2].strip("\""))   #I do not want the quotations, will strip them
else:
   print(f"The given command is invalid and the error is {err.splitlines()}")
```

16.0.2

--Working with text files

--Reading or writing into the text files

**We have 3 methods

Create a new file

Write into the existing file

Read from the existing file/file

**We have to open the file into the W mode(Write)
a mode(append)
r mode(read)

-Based on the cusrsor only I am ariting data into the file, where the cursor is now.

-When we will close in python, automatically data will be saved.

-Always you should get the cursor position like

```
    #creation of empty file
    fo = open("newfile.txt", "w")

    fo.close()
```

```
#Reading or writing data into the text files
#create an empty file
#Always we should have cursor position
'''fo = open("newfile.txt", "w") #fo is the file object will give us the cursor position
#Here I am creating an empty file
#I want to see in which mode the file is opened.
print(fo.mode) #This will give us the mode.
fo.close()
'''
#I want to create a new file and write ito it

fo = open("random.txt", "w")
#If I want to add second line, we need to add \n on every line, so the cursor will go to next line.
fo.write("This is a test file \n")
fo.write("This is the second line \n")
fo.close()
```

```
This is a test file
This is the second line
```

```
#Reading or writing data into the text files
#create an empty file
#Always we should have cursor position
'''fo = open("newfile.txt", "w") #fo is the file object will give us the cursor position
#Here I am creating an empty file
#I want to see in which mode the file is opened.
print(fo.mode) #This will give us the mode.
fo.close()
```

'''
#I want to create a new file and write ito it
my_content = ["This is data1 \n", "This is data2 \n", "This is data3"]
fo = open("random.txt", "w")
#If I want to add second line, we need to add \n on every line, so the cursor will go to next line.
'''fo.write("This is a test file \n")
fo.write("This is the second line \n")'''
#To Write multiple lines, we do not need to add multiple line s in code, we can use writelines
fo.writelines(my_content)
fo.close()
#When we will open a file in "w" mode, then python will simply remove this file and create a new file.

##Lets try to write files with the help of loop

my_content=["This is data1", "this is data2", "This is data3", "This is data4"]

fo = open("new-random.txt", "w")
for i in my_content:
    fo.write(i+"\n")
fo.close()

This is data1
this is data2
This is data3
This is data4

#Want to open the file in append mode
my_content=["This is data1", "this is data2", "This is data3", "This is data4"]

fo = open("new-random.txt", "a")
for i in my_content:
    fo.write(i+"\n")
fo.close()

```
This is data1
this is data2
This is data3
This is data4
This is data1
this is data2
This is data3
This is data4
-----------------------
#Want to open the file in append mode
my_content=["This is data1", "this is data2", "This is data3", "This is data4"]

fo = open("new-random.txt", "a")
for i in my_content:
    fo.write(i+"\n")
fo.close()
#it will not disturb the existing data and the new data will be appended at the end.
#if the file is not there then append mode will create a new file and add the data.
-----------------------
#I want to read the contents of the file.
fo = open("new-random.txt", "r")
contents = fo.read()
print(contents)
fo.close()
-----------------------
This is data1
this is data2
This is data3
This is data4
This is data1
this is data2
This is data3
This is data4
-----------------------
#I want to read data line by line
```

```
fo = open("new-random.txt", "r")
contents = fo.readline()
print(contents)
fo.close()
----------------------
This is data1
-----------------------
#If I want to read all the data and print the data into a list

fo = open("new-random.txt", "r")
contents = fo.readlines()
print(contents)
fo.close()
#If I want to print only the first 3 lines(we need to take the index value)
for i in range(0,3):
   print(contents[i])
-------------------------
['This is data1\n', 'this is data2\n', 'This is data3\n', 'This is data4\n', 'This is data1\n', 'this is
data2\n', 'This is data3\n', 'This is data4\n']
This is data1

this is data2

This is data3
-------------------------
#If I want to read all the data and print the data into a list

fo = open("new-random.txt", "r")
contents = fo.readlines()
print(contents)
fo.close()
#If I want to print only the first 3 lines(we need to take the index value)
for i in range(0,3):
   print(contents[i])
#If I want to print the last line
```

print(contents[-1])

This is data4

--Copy the content of a file from a source file to the destination file
---
#Copy the content of a source file into the destination file
#When trying to read or write into a file, please give the complete path
#sfn = "/Users/ASUTOSH/COURSES/python-scripting-automation/new-random.txt"
#dfn = "/Users/ASUTOSH/COURSES/python-scripting-automation/random-new.txt"
#I want to read the source file and the destination file location while running the script.
sfn = input("Enter the source file destination. Give full path: ")
dfn = input("Enter the destination file location.Give the fulll path: ")

sfo = open(sfn,"r")
contents = sfo.read()
sfo.close()

dfo = open(dfn,"w")
dfo.write(contents)
dfo.close()
#If we will open the file without any mode, so by default it will be in read mode.

--Working with CSV-----

First of all, what is a CSV ?

CSV--> CommaSeparatedValues

It is a simple file format used to store tabular data, such as a spreadsheet/excel or database.

CSV is a convenient way to export data to spreadsheets and databases as well as import or use it in other programs.

For Example---

S_NO,Name,Salary,Skill
1,Asutosh,2000,Python

```
#If I want to read a csv file
import csv
req_file = input("Enter the csv file path: ")
f = open(req_file, "r")
csv_reader = csv.reader(f,delimiter="|") #If the separator in the csv file is other than the ","
for i in csv_reader:
    print(i)

f.close()
```

```
Enter the csv file path: /Users/ASUTOSH/COURSES/python-scripting-automation/new_file.csv
['S_No', 'Name', 'Salary', 'Skill']
['1', 'Asutosh', '2000', 'Python']
['2', 'Ankita', '3000', 'Bash']
```

How to get header and no of rows in a csv file

```
#How to get header and no of rows in a csv file
import csv
req_file = input("Enter the csv file path: ")
f = open(req_file, "r")
csv_reader = csv.reader(f,delimiter="|") #If the separator in the csv file is other than the ","
#print(f"The header is: \n {list(csv_reader)[0]}")
#Else we can use next function, so that it will move the cursor to the next line.
header = next(csv_reader)
print(header)
#If only want header, no need for loop
```

```
#Then if we need the number of rows after the header
print(f"The number of rows are : {len(list(csv_reader))}")
#Now in csv_reader there is only two rows.
#for i in csv_reader:
    #print(i)

f.close()
```

```
-----------------------------------------
['S_No', 'Name', 'Salary', 'Skill']
['1', 'Asutosh', '2000', 'Python']
['2', 'Ankita', '3000', 'Bash']
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/intro-to-csv-file.py
Enter the csv file path: /Users/ASUTOSH/COURSES/python-scripting-automation/new_file.csv
['S_No', 'Name', 'Salary', 'Skill']
2
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/intro-to-csv-file.py
Enter the csv file path: /Users/ASUTOSH/COURSES/python-scripting-automation/new_file.csv
['S_No', 'Name', 'Salary', 'Skill']
The number of rows are : 2
-------------------------------------------------------------------------
```

```
--Creating CSV file
-------------------
#create csv file using python
import csv
req_file = "/Users/ASUTOSH/COURSES/python-scripting-automation/demo.csv"
fo = open(req_file, "w",newline="") #To delete the empty line after every entry.
csv_writer = csv.writer(fo,delimiter=",")
#csv_writer.writerow(["S_NO", "Name", "Salary", "Skill"])
#When we are going to add the lines one by one or got new data to add, we can use above
operations
```

```
#Now instead of writing every line, we can enter the data in list of lists
data = [["SL_No","Name","Salary","Skill"],["1","Asutosh","3000","Python"]]
csv_writer.writerows(data)

fo.close

----------------------
SL_No,Name,Salary,Skill
1,Asutosh,3000,Python
-------------------------------------------------------------------------
--Working with Json Files
------------------------
 JSON:
JSON (JavaScript Object Notation) is a popular data format used for representing structured
data.

It's common to transmit and receive data between a server and web application in JSON format.

PYTHON                JSON EQUIVALENT
-------              ----------------
dict                   object      <=======Dictionary is always an object for the json.

list,tuple               array

str                    string

int,float,int             number

True                    true

False                   false

None                    null

------------------
```

```
import json
'''req_file = input("Enter the path of the json file: ")
fo = open(req_file, "r")
#print(fo.read()) #We can also run fo.read to get the normal output in json format
print(json.load(fo)) #We can process json data because that is string, so we load json data and that will convert it into a "Dictionary".



fo.close()'''
#How I can store data into a json file
#When we are processing json data as a distionary, so we should have our data as a dictionary to insert the data into a json file.

my_dict = {"Name": "Asutosh", "Skills":["Python","Shell","K8s"], "Salary": 3000}
req_file = input("Enter the file name: ")
fo = open(req_file, "w")
json.dump(my_dict,fo,indent=4) #we are just dumping the dict data into the file.

fo.close()
```

```
{
    "Name": "Asutosh",
    "Skills": [
        "Python",
        "Shell",
        "K8s"
    ],
    "Salary": 3000
}
```

--Eception Handling

Exception Handling:

A Python program terminates as soon as it encounters an error.
Error can be a syntax error or an exception.

Syntax errors are nothing but incorrect statements.
(No way to handle syntax errors)

An exception is an error that happens during execution of a program. So exceptions are runtime errors.

When runtime error occurs then python generates an exception that can be handled, which avoids our program/script to terminate or crash.

When to use Exception handling:
When you think that you have a code which can produce an error then you can use exception handling.

How to handle Exceptions?

Exceptions can be handled using try and except statement.

```
try:

    Run this code

except:

    Run this code if an exception occurs
```

- Exceptions Examples:
IndexError
ZeroDivisionError
ImportError
ValueError
TypeError
NameError

```
FileNotFoundError
IOError

--------------------------------
#print("Welcome to exceptions concept") #If I will do a mistake in the first line, will not get the output for 2nd line.
#print("Now it is fine")
#print(4/0)
#Basically errors are two types, syntax error and exception error
'''  File "/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py", line 3
    print(4/0)
    ^
SyntaxError: invalid syntax <================
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$ /usr/local/bin/python3 /Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py
Welcome to exceptions concept
Now it is fine
Traceback (most recent call last):
  File "/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py", line 3, in <module>
    print(4/0)
ZeroDivisionError: division by zero <================
these errors are called runtime errors====> exception error'''
#We can handle the runtime errors/exceptions

'''fo = open("newfile.out", "r")#So here the file is not present so "FileNotFound Error" Exception
fo.read()
fo.close()'''
#We can run above code in below format, SO that the program will not terminate with error
'''try:                 #If the file is there then the (try) block will execute
    fo = open("newfile.out", "r")
    print(fo.read())
```

```
    fo.close()
#except :              #If file is not there then except block will run.
    #print("There is some problrem with the file")
#Or we acn also print the exception error with except
except Exception as e:
    print(e)
'''
#Try to print my_list
'''
my_list = [1,2,3,4,5]
#print(my_list[5])
#here we are getting "index out of range" error, because there is no 5th index.
try:
    print(my_list[5])
except Exception as e:
    print(e)
#After getting error also the next line will executes
print("This line will also exceutes")'''
#I want to import a module called fabric
try:
  import fabric
except Exception as e:
    print(e)

#here we are getting import error.ModuleNotFoundError: No module named 'fabric'


----------------
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py
list index out of range
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py
list index out of range
```

```
This line will also exceutes
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py
Traceback (most recent call last):
  File "/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py",
line 45, in <module>
    import fabric
ModuleNotFoundError: No module named 'fabric'
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py
No module named 'fabric'
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/working-with-exceptions.py
No module named 'fabric'
--------------------------------------------------------------------------------
--Exception Handling for Known Exceptions
---------------------------------------
#Exception Handling for Known Exceptions
#print(a)
#here we will get Name Error NameError: name 'a' is not defined

#print(4+"string")
#here we will get type error. TypeError: unsupported operand type(s) for +: 'int' and 'str'

#fo = open("file.out")
#Here we are getting file not found. FileNotFoundError: [Errno 2] No such file or directory:
'file.out'

#print(5/0)
#here we will get zero division error
# to avoid or to handle the known runtime errors we can do below
#here we know the types of errors, so we can handle directly.
```

```
try:
    print(a)
    #print(4+"string")
    #open("file.out")
    #print(5/0)
    import fabric
except NameError:
    print("The value is not defined.")
except TypeError:
    print("not possible")
except FileNotFoundError:
    print("File not found")
except ZeroDivisionError:
    print("Not possible with zero division")
except ImportError:
    print("No module")
except Exception as e:
    print(e)
#So if no exception is mentioned then it will go to the last line and print the error.

#If two lines are enabled under the try: and first line itself will give the exception error, then the next line will not exceute.
finally:
    print("finally this block will execute")
#It means after trying with all the blocks finally final block will execute.
-------------------------------------------------------------------------------
--Difference between try except finally and try except else
-------------------------------------------------------
try:
    #a =10
    print(a)
except NameError:
    print("value not found")
except Exception as e:
    print(e)
```

```
#whether exception is there or not, whatever the code is there in finally is going to exceute.
else:
   print("This will execute if there is no exceptions in try block")

finally:
   print("this will execute always")

---------------------------------------
10
This will execute if there is no exceptions in try block
this will execute always
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/exception-difference-finally-else.py
value not found
this will execute always
-------------------------------------------------------------------------------------
-Raise user Defined Exceptions
-----------------------------

We can create custom Exceptions using:

    raise =====>(Used to raise an existing exceptions)

    assert =====> (Used to create an AssertionError)

---------------------------------
#How to raise userdefined exceptions
'''raise Exception("This is an Exception")

raise NameError("Value not defined")
'''
#for example
'''age = 25
if age > 30:
```

```
    print("valid Age")
else:
    raise ValueError("Age is less than 30")
#this is the way we can raise the existing exception
'''
#Another method is assert is used to create assertion error.
#assert will raise error if the condition written inside is false.
#assert (4<3)
#assert (4>3)

age = 31

try:
    assert age<30
    print("Valid age")
#If want to handle the assertion error
except AssertionError:
    print("Raised with assert because age is greater than 30")
except:
    print("exception occured")
--------------------------------
exception occured
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/custom-exceptions.py
Valid age
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/custom-exceptions.py
exception occured
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$
/usr/local/bin/python3
/Users/ASUTOSH/COURSES/python-scripting-automation/custom-exceptions.py
Raised with assert because age is less than 30
```

(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$ /usr/local/bin/python3 /Users/ASUTOSH/COURSES/python-scripting-automation/custom-exceptions.py
Valid age
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$ /usr/local/bin/python3 /Users/ASUTOSH/COURSES/python-scripting-automation/custom-exceptions.py
Raised with assert because age is greater than 30

--FUNCTIONS

Introduction to Functions

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.

For example----

```
#write a program that will clear the screen and also give us the list of the files as per the platform
import os
import time
import platform

def mycode(cmd1,cmd2):
    print("Clearing the screen.....")
    time.sleep(2)
    os.system(cmd1)
    print("Listing directories and files....")
    time.sleep(2)
    os.system(cmd2)
```

```
if platform.system == "Windows":
   mycode("cls","dir")
else:
   mycode("clear","ls -lrt")
#here no need to run the code, again and again. We can use a function and can call it anytime.
----------------
Clearing the screen.....
Listing directories and files....
(base) ASUTOSH-MACBOOK-PRO:python-scripting-automation ASUTOSH$ ls -lrt
total 496
-rw-r--r--  1 ASUTOSH  admin   129 Aug 17 23:12 hello-world.py
-rw-r--r--  1 ASUTOSH  admin    88 Aug 18 04:15 variable.py
-rw-r--r--  1 ASUTOSH  admin   425 Aug 18 05:13 multiple-variables.py
-rw-r--r--  1 ASUTOSH  admin   480 Aug 18 05:14 datatype.py
-rw-r--r--  1 ASUTOSH  admin   493 Aug 19 03:59 input-output.py
-rw-r--r--  1 ASUTOSH  admin   913 Aug 19 04:42 strings.py
-rw-r--r--  1 ASUTOSH  admin   257 Aug 19 04:52 boolean-result.py
-rw-r--r--  1 ASUTOSH  admin   463 Aug 19 05:17 join-center-zfill[zero-fill].py
-rw-r--r--  1 ASUTOSH  admin   502 Aug 19 23:33 strip-split-operations.py
------------------------------------------------
--How to define a Function and use defined Function ?
------------------------------------------------
Why Function are used ?
Code Reusability
Improve Modularity

Types of Functions:
Basically, we can divide functions into the following two types:

Built-in Functions:
      Functions that are built into Python.

User-Defined Functions:
      Functions defined by the users themselves
```

```
-----------------
def display():
    print("This is first line")
    print("this is the second line")
    return none

display()

display()
#We can call the function anytime.

x = [1,2,3,4,5,6]
print(len(x))    #This(len) is a python built-in function/default function.
#We can call the function anytime.
----------
This is first line
this is the second line
This is first line
this is the second line
------------------------------------------------------------
-- Functions with arguments and return value
-------------------------------------------
 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

#Convert simple codes into function.
def welcome_messgae():
    print("Welcome")
    return None
def known_concepts():
```

```
    print("Now we are good with basics")
    return None
def new_concepts():
    print("Functions")
    return None

#When we will not call the functions, we will not get any display.
welcome_messgae()
-------------------------------------------------------
--Calling a function from another function and scope of the variables
-------------------------------------------------------------------

#calling a function from another function

def myfunction1():
    x =10       #here the value of x can be used by only myfunction1, this is called local variable.
    print("Welcome to functions")
    print("The value is", x)
    #myfunction2()
    return None

def myfunction2(y): #here the value passed to the function is called parameter.
    print("Thank you!!")
    print("The value is", y)
    return None
#sometimes we need to call one function in another function.
#x = 10 #So here the value can be access by both the function, called scope of variable. This is
global variable.
#myfunction1()

#lets say I am writing another function called main and I am calling function1 from main

def main():
    #global x
    x =20 #here x is not global to function2, it is local to main()
```

```
    myfunction1()
    #there is another way to pass the value to function2
    myfunction2(x)  #here the value passed to the function is called argument.
    #Now this variable x and the variable y are diferent.
    return None

main()
-----------------------------------------------------------------------
--Simple Functions with Arguments
---------------------------------
'''#num = eval(input("Enter the number"))
def get_results(num):
    result = num + 10
    print("The result is:", result)
    return None

input = eval(input("Enter the number"))
get_results(input)'''

def get_addition(a,b):
    aresult = a + b
    print("The addition result is: ", aresult)
    return None
def main():

    x = eval(input("Enter the first number"))
    y = eval(input("Enter the second number"))
    get_addition(x,y)
    return None

main()

-----------------------------------------------------------------------
--Functions with arguments and return value
------------------------------------------
```

```
def main():
  a = eval(input("Enter the number"))
  b = eval(input("Enter the number"))
  result = addition(a,b) #here the return value from the function will be saved inside the variable
  print("The sum value of two numbers are:", result)
  return None

def addition(a,b):
   sum = a + b
   #print("The addition of two numbers are: ", sum) #Lets say I do not want to print the sum here
   #I will print the sum in main function. So I need to return the value of sum
   return sum #We will use none, where we do not need to return any value.

main()
-----------------------------------------------------------------
--Functions with default arguments
-------------------------------
def display(a=10):
   print("The value of a is:" , a)
   return None
display() #This function will need one argument.If we are not passing a value here, we need to
pass default value
display(6)

-----------------------
The value of a is: 10
The value of a is: 6
-----------------------------------------------------------------
--Functions with keyword-based arguments
-------------------------------------
#Function with keyword-based arguments
def display(a,b):
   print(f"a = {a}")
   return None
```

```
display(3,4)
display(a=3,b=4)
display(a=4,b=3)
-----------------
a = 3
a = 3
a = 4
-----------------------------------------------------------------------
-- Functions with Variable length arguments
----------------------------------------
#Functions with variable length arguments
def display(*arg):  #we can do that by adding *arg
    print(arg)
    return None
display() #This function will need one argument.If we are not passing a value here, we need to
pass default value
display(6,7,8)
#generaly we can not give arguments more than the parameters assigned. But we can do that

-----
()
(6, 7, 8)
-----------------------------------------------------------------
-- How to use Functions of one script into another script, what is __name__ ? and how to create
user defined modules
-----------------------------------------------------------------------------------------------------------
---
****Script1
------------def addition(a,b):
    result = a + b
    print("The result is :" , result)
    return None
def subtraction(a,b):
    print(f"The subtraction of two numbers is {a-b}")
    return None
```

```
def main():
  x = 9
  y = 10
  addition(x,y)
  subtraction(x,y)
  return None

if __name__ == "__main__": #So while we are importing script1 in script2, the __name__ changed to script1, so the if condition is not true, so it will not exceute the main function. So we will not get the output from the script1.
    main()
#So now if we will run script2, we will only get the output of script2, no script1.
#we need to use functions from script1 in another script
#print("This is script1",__name__)
-----------------------------------
******Script2
-------------
#We can import the functions from script1 in script2.
#But we need to comment out the x and y value in script1, because python will run those values also.
import script1

def mult(a,b):
    print("Result is ",a*b)
    return None

def main():
    x = 10
    y = 20
    mult(x,y)
    script1.addition(x,y)
    return None
if __name__=="__main__":
     main()
```

#Now if I want to see only the value of script2 and do not want to see script1 output.
#import script1
#print("This is script2",__name__)
#value
#script1
#This is script2 __main__
#So if we are printing "__name__" on script1, instead of "__main__" we are getting only script1

*****We will folow this method

```
import sys
import os
import time
def addition(a,b):
    print("The result is", a+b)
    return None
def main():
    x = 20
    y = 40
    addition(x,y)
    return None

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

#This is the user defined modules we are creating, by this main function will not execute, if we will import the script in other script.
#Moving forward we are going to write our python script in this format.

--Simple exception handling to changing current working directory
---
'''import os
print(os.getcwd())
print(os.chdir("/Users/ASUTOSH/COURSES/python-scripting-automation/test.txt"))

```
print(os.chdir("/Users/ASUTOSH/COURSES/python-scripting-automation/hello.txt"))'''
#So while change directory we have below usuall execptions, need to write a python script
#NotADirectoryError:
#permission denied
#file not found
import os
'''req_path = input("Enter the required path to change the directory: ")
print("The current working directory is : ", os.getcwd())
try:
    os.chdir(req_path)
    print("Now your new working directory is:", os.getcwd())
except PermissionError:
    print("The given path is not a valid path")
except FileNotFoundError:
    print("The given path is not a valid path")
except NotADirectoryError:
    print("The given path is not a 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()

-------------------------
Enter path to change working dir:
/Users/ASUTOSH/COURSES/python-scripting-automation/ankita
The current working dir is:  /Users/ASUTOSH/COURSES/python-scripting-automation
Given path is not a valid path. So cant change working directory

-----------------------------------------------------------------------------------
#Find the repeatation of the odd numbers given in a string
from collections import Counter
x = "1 2 3 4 5 6 7 8 9 0 7 6 5 4 3 2 1 3 5 7 9 1 3 5 7 "
y = list(x.split())
print(y)
o_l=[]
for i in y:
   if int(i) % 2 != 0:
      o_l.append(i)
print(o_l)
#count repeated elements in the list
c = Counter(o_l)
print(c)
-----------------------------------
['1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '7', '6', '5', '4', '3', '2', '1', '3', '5', '7', '9', '1', '3', '5', '7']
['1', '3', '5', '7', '9', '7', '5', '3', '1', '3', '5', '7', '9', '1', '3', '5', '7']
Counter({'3': 4, '5': 4, '7': 4, '1': 3, '9': 2})
-----------------------------------------------------------------------------------------------------------------------
----------
--Regular Expression with re Module
-----------------------------------
--Introduction to regular expressions
-----------------------------------
```

**What is 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.

e.g--

"i[st]" ----> is,it

"python[23]" -----> python2,python3

If I will use "python[23]" inside a RegEx, then it will represent two words. python2,python3

We can split strings based on the srings

e.g -

```
>>> my_str = "python is a good language and it is very easy to learn"
>>> my_list = my_str.split("is")
>>> my_list
['python ', ' a good language and it ', ' very easy to learn']
>>> my_list = my_str.split("easy")
>>> my_list
['python is a good language and it is very ', ' to learn']
```

---If want to use regular expressions need to import re module

In normal string you can not split the string based on two string. With regex(re) we can do that

```
>>> import re
```

Here I can split the string with "is" and "It"

```
>>> re.split("i[st]",my_str)
['python ', ' a good language and ', ' ', ' very easy to learn']
```

--Basic Rules to create a pattern for regex part1

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” <-----This a pattern

“Python[23]” ====> “Python2” “Python3”

**python[23] is a pattern

r“Python” or r”Python[23]”

Learning pattern creation with findall operation:

How to use findall ?

import re

print(re.findall(pattern,text))

```
import re
text = "This is a python and it is easy to learn"
#This will match the string, with the given pattern. like is with first "Th", then "is" with "hi", then "is" with "is" like this
my_pat = "is" #This is the pattern
print(len(re.findall(my_pat,text))) #Now if we want to find out the lenth of the patern, like the number of times the pattern is there
print(re.findall(my_pat,text))

#Now I want to search for "is" and "it" in the string, so I need to make a pattern, so that I can search
#"is" "it" here i is common, so I can write i[st]
new_pat = "i[st]"
print(re.findall(new_pat,text))

new_pat1 = "p"
print(re.findall(new_pat1,text))
```

```
3
['is', 'is', 'is']
['is', 'is', 'it', 'is']
['p']
```

--Rules to Create 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 .

```
new_pat1 = "[@as]" #EIther a or b or c
print(re.findall(new_pat1,text))
```

```
['s', 's', 'a', 'a', 's', 'a', 's', 'a']
```

```
new_pat1 = "[a-dx-zh-l]" #EIther the string in a to d, or x-z or h-l
print(re.findall(new_pat1,text))
```

```
['h', 'i', 'i', 'a', 'y', 'h', 'a', 'd', 'i', 'i', 'a', 'y', 'l', 'a']
```

```
new_pat1 = "\w" #Matches any single letter, digit or underscore
print(re.findall(new_pat1,text))

#We are going to get all the strings, except space
```

```
['T', 'h', 'i', 's', 'i', 's', 'a', 'p', 'y', 't', 'h', 'o', 'n', 'a', 'n', 'd', 'i', 't', 'i', 's', 'e', 'a', 's', 'y', 't', 'o', 'l', 'e', 'a', 'r', 'n']
```

**If I will write two \w\w, it will look for the pattern 'aa' or 'bb' or two letter string

```
new_pat1 = "\w\w" #If I will write two \w\w, it will look for the pattern 'aa' or 'bb' or two letter string
print(re.findall(new_pat1,text))
```

```
['Th', 'is', 'is', 'py', 'th', 'on', 'an', 'it', 'is', 'ea', 'sy', 'to', 'le', 'ar']
```

```
new_pat1 = "." #matches any single character except newline character
print(re.findall(new_pat1,text))
```

```
['T', 'h', 'i', 's', ' ', 'i', 's', ' ', 'a', ' ', 'p', 'y', 't', 'h', 'o', 'n', ' ', 'a', 'n', 'd', ' ', 'i', 't', ' ', 'i', 's', ' ', 'e', 'a', 's', 'y', ' ', 't', 'o', ' ', 'l', 'e', 'a', 'r', 'n']
```

```
new_pat1 = "\W" #Matches any character not part of \w
print(re.findall(new_pat1,text))
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

```
new_pat1 = "\d" #Matches decimal digit 0-9
print(re.findall(new_pat1,text))

--------------
[]    #There is no decimal numbers.

--------------
text = "This is a python and it is easy.@_---- to learn_@@@@_____@@@@  ....@"

new_pat1 = "." #matches any single character except newline character
print(re.findall(new_pat1,text))

--------------
['T', 'h', 'i', 's', ' ', 'i', 's', ' ', 'a', ' ', 'p', 'y', 't', 'h', 'o', 'n', ' ', 'a', 'n', 'd', ' ', 'i', 't', ' ', 'i', 's', ' ', 'e', 'a', 's', 'y',
'.', '@', '_', '-', '-', '-', '-', ' ', 't', 'o', ' ', 'l', 'e', 'a', 'r', 'n', '_', '@', '@', '@', '@', '_', '_', '_', '_', '@', '@', '@',
'@', ' ', ' ', '.', '.', '.', '.', '@']

--------------
#Now with ".."

new_pat1 = ".." #matches any single character except newline character
print(re.findall(new_pat1,text))

-------------
['Th', 'is', ' i', 's ', 'a ', 'py', 'th', 'on', ' a', 'nd', ' i', 't ', 'is', ' e', 'as', 'y.', '@_', '--', '--', ' t', 'o ', 'le', 'ar',
'n_', '@@', '@@', '__', '__', '@@', '@@', '  ', '..', '..']

--------------
#with "..."

new_pat1 = "..." #matches any single character except newline character
print(re.findall(new_pat1,text))

--------------
```

['Thi', 's i', 's a', ' py', 'tho', 'n a', 'nd ', 'it ', 'is ', 'eas', 'y.@', '_--', '-- ', 'to ', 'lea', 'rn_', '@@@', '@__', '__@', '@@@', ' .', '...']

#Now I want to match the "." with "." only

new_pat1 = "\." #matches any single character except newline character
print(re.findall(new_pat1,text))

['.', '.', '.', '.', '.']

-For Example-----

#Lets say I want to get the IP only
pattern = "\d\d\d.\d\d\d.\d\d\d.\d\d\d" #here wit this option I am getting all the other numbers also
#If I want to get only the Ip addressthen we have to use "\."
print(re.findall(pattern,text))

['255.100.102.103', '2345678909876 54']

---
text = "This is my ip of a db server: 255.100.102.103 23456789098765 4321"
#Lets say I want to get the IP only
pattern = "\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d" #here wit this option I am getting all the other numbers also
#If I want to get only the Ip addressthen we have to use "\."
print(re.findall(pattern,text))

['255.100.102.103']

--Rules to create a pattern part-2

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

** $ - End of the string (Newline character in-case of multiline string)

** \b - Empty string at the begining or end of the word

** \B - Empty string not at the end or begining of the word

** \t, \n, \r - Matches tab, newline, return respectively.

```
new_pat1 = "^This" #Matches Start of the string(and start of the line in-case of multiline string)
print(re.findall(new_pat1,text))
```

```
['This']
```

```
new_pat1 = "@$" #End of the string (Newline character in-case of multiline string)
print(re.findall(new_pat1,text))
```

```
new_pat1 = "\\blearn" #Empty string at the begining or end of the word
print(re.findall(new_pat1,text))
```

\b for back slash, so need to add one more \ here.

```
['learn']
```

Instead of \ we can use "r" now the \b is going to use as space.

```
new_pat1 = r"\bis" #Empty string at the begining or end of the word
print(re.findall(new_pat1,text))
```

```
['is', 'is']
```

```
text = "  This is a python and itis easy.@_---- to learn_@@@@_____@@@@  ....@ "
```

```
new_pat1 = r"\bis\b" #Empty string at the begining or end of the word
print(re.findall(new_pat1,text))
```

```
['is']
```

Here this will check if there is any space before and after the string

---

```
new_pat1 = r"\Blearn" #Empty string not at the end or begining of the word
print(re.findall(new_pat1,text))
```

```
[]
```

```
new_pat1 = r"\n" #Matches tab, newline, return respectively.
print(re.findall(new_pat1,text))
```

```
['\n']
```

--Rules to create a pattern part 3

** {2} - Exactly two times ----- We can check this for any character.

** {2,4} - two , three or four times

** {2,} - two or more times

** + - one or more

** * - zero or more times

** ? - Once or None

---
new_pat1 = "@{4}" #Matches tab, newline, return respectively.
print(re.findall(new_pat1,text))

['@@@@', '@@@@']

---
new_pat1 = r"\bpytho{1,2,4}n\b" #Matches tab, newline, return respectively.
print(re.findall(new_pat1,text))

--Rules to create pattern part 4

-Regex with Flags

***Simple Flags for Regex

Abbreviation	Full Name	Description
re.I	re.IGNORECASE	Makes the regular expressions casen censitive.

re.L re.LOCALE The behavior of some special sequences like \w,\W,\b,\s,\S will be made depandant on the current locale, i.e - user's language and country also.

re.M re.MULTILINE ^ and $ will match at the beginning and at the end of each line and not just at the beginning and the end of the string

re.S re.DOTALL The "." will match every character plus the new line

re.U re.UNICODE Makes \w,\W,\b,\B,\d,\D,\s,\S dependent on Unicode character properties.

re.X re.VERBOSE Allowing "verbose regular expressions", i.e white space are ignored. this means that spaces,tabs and carriage returns are not matched as such. If you want to match a space in a verbose regular expression, you will need to escape it by escaping it with a backslash in front of it or including it or include it in a character class. # are also ignored except when in a character class or preceded by a non escaped backslash. Everything following a "#" will be ignored until the end of the line,so this character can be used to start a comment.

****To specify more than one of them, use | operator to connect them.

for example--

re.search(pattern,string,flags=re.IGNORECASE|re.MULTILINE|re.UNICODE)

--Introduction to OOPS

-What is OOPs

It is about class,object,inheritance,data encapsulation, data abstraction,polimorphism.

OOPs is all about the use of class and object.

-What is Object

Object could be anything which exists in realtime, like human,fan,car, email, any application, jenkins,weblogic,tomcat, apache

Each object has some characteristics and functions.

-We are using oops to create object

**WHy OOPs

-Why we create objects , to group related functions.

-To create a template/blueprint

-OOps is a concept where characteristic and functions of a real-life object is packaged as a single entity in the code.

--Class and object attributes

-Class is a template/blueprint to create an object.

- Class is the combination of attributes and methods.

- We can define attributes for class and objects. Means for class level we can define some variables and for object level we can define some variables.

```
class employee:
    #If we want to count the number of employees.
```

```
    count = 0 #This is class attribute.
    def get_emp_age_sal_name(self,name,age,sal): #We are creating a method to get the
employee details
        self.name = name    #These are class object attributes
        self.age = age
        self.sal = sal
        #if we want to call the display function here itself to display the data.
        #self.display_data()
        #We can add the employee count in the method itself to call the method employee_count.
        self.employee_count()
    def employee_count(self):
        employee.count = employee.count+1
    def display_data(self):
        print("The name of the employee is", self.name)
        print(f"The age of the employee is {self.age}")
        print(f"The salary of the employee is {self.sal}")

emp1 = employee()

emp2 = employee()
#we are storing the whole class template in the object emp1 and emp2.
emp1.get_emp_age_sal_name("Asutosh",33,50000)
#emp1.employee_count()
emp2.get_emp_age_sal_name("Ankita",26,50000)
#emp2.employee_count()
emp1.display_data()
emp2.display_data()
print(employee.count)
---------------------------
The name of the employee is Asutosh
The age of the employee is 33
The salary of the employee is 50000
The name of the employee is Ankita
The age of the employee is 26
The salary of the employee is 50000
```

2

--Constructor of a class

-A constructor is a special method that is called by default whenever you create an object from a class.

-----It is called initialization method, and it will be called automatically, whenwver you will call the object of the class.

```
class emp:
    count=0
    def __init__(self):  #This will be called automatically whenever the object will be initialized.
        print("This is a constructor")
        #lets add emp count in the constructor.
        #emp.count = emp.count+1
    def get_employees(self):
        print("This is a display function")

emp1=emp()
emp2=emp()
#emp1.get_employees()
#emp2.get_employees()
```

```
This is a constructor
This is a constructor
```

```
class emp:
    count=0
    def __init__(self):  #This will be called automatically whenever the object will be initialized.
        #print("This is a constructor")
        #lets add emp count in the constructor.
        emp.count = emp.count+1
    def get_employees(self):
```

```
        print("This is a display function")

emp1=emp()
emp2=emp()
print("The employee count is: ", emp.count) #There is no object created, so no output.After creating object
#emp1.get_employees()
#emp2.get_employees()

-------------------------
The employee count is:  2
--------------------------
--One more method
-------------------
    **usage-of-constructor.py

    class employee:
    def __init__(self,name,salary):
        self.name = name
        self.salary = salary
    def display(self):
        print("The name is :", self.name)
        print("The salary is:", self.salary)

#Now if we will create object, the __init__ method will call automatically, so it needs arguments to pass.So we have to pass the arguments while creating objects.
emp1=employee("Asutosh",90000) #here we are assigning arguments to the method.
emp2 = employee("Abhinash",100000)
emp1.display()
emp2.display()
-----------------------------------
The name is : Asutosh
The salary is: 90000
The name is : Abhinash
The salary is: 100000
```

```
-------------------------
class employee:
    def __init__(self,name,salary):
        self.name = name
        self.salary = salary
        self.display() #we are calling the method inside the constructor
    def display(self):
        print("The name is :", self.name)
        print("The salary is:", self.salary)

#Now if we will create object, the __init__ method will call automatically, so it needs arguments to pass.So we have to pass the arguments while creating objects.
emp1=employee("Asutosh",90000) #here we are assigning arguments to the method.
emp2 = employee("Abhinash",100000)
#emp1.display()
#emp2.display()
#Now if we will call the method display() inside the method __init__ itself, then we do not need to call the method
-----------------------------------------
The name is : Asutosh
The salary is: 90000
The name is : Abhinash
The salary is: 100000
```