In [3]:
```
age=18
if age > 18:
 print("Person is an adult")
elif age == 18:
 print("Person is adulting")
else:
 print("Kiddo !!!")
```

```
Person is adulting
```

In [9]:
```
'''
    Single line comments using #
    Multiple line comments in python --- Three single or double quotes opening and closing

'''
num1 =15  #integer
num2 =10.50 # float
num3 = 10+5j #complex
print(type(num1))
print(type(num2))
print(type(num3))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

In [10]:
```
#String
firstName="Virat"
lastName='Kohli'
print(type(firstName))
print(type(lastName))
```

```
<class 'str'>
<class 'str'>
```

In [19]:
```
#List
numbers=[10,11,9,5,12,1,2,3,4,5]
print(type(numbers))
listOfnumbers=[10,11.5,9.6,0.5,12,1,2,3,4,5]
print(type(listOfnumbers))
listOfList=[10,11.5,9.6,numbers]
print(type(listOfList))
listOfLists=[10,11.5,9.6,[1,2,3,4,5,6]]
print(type(listOfLists))
```

```
listOfMixDataType=[10,"Hello",'Good',15,50,numbers]
print(type(listOfMixDataType))
print(listOfMixDataType)
emptyList=list()
print(type(emptyList))
print(emptyList)
```

```
<class 'list'>
<class 'list'>
<class 'list'>
<class 'list'>
<class 'list'>
[10, 'Hello', 'Good', 15, 50, [10, 11, 9, 5, 12, 1, 2, 3, 4, 5]]
<class 'list'>
[]
```

In [20]:
```
emptyList=[]
print(type(emptyList))
print(emptyList)
```

```
<class 'list'>
[]
```

In [29]:
```
            #0  1 2 3 4
listOfNumbers=[1,2,3,4,5] #index starts with 0 and ends with length-1
print(listOfNumbers[2])
print(len(listOfNumbers))
print(listOfNumbers.__doc__)
print(dir(listOfNumbers))
```

```
3
5
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list.
The argument must be an iterable if specified.
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq_
_', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__imul_
_', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce_
_', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__sub
classhook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

In [38]:
```
listOfNumbers=[1,2,3,4,5,1,15]
print(listOfNumbers.__len__)
print(listOfNumbers.count)
print(listOfNumbers.count(1)) #Count function or method gives you the count of specific element or item present in the lis
```

```
<method-wrapper '__len__' of list object at 0x000002A0B70B0BC0>
<built-in method count of list object at 0x000002A0B70B0BC0>
2
```

In [43]:
```
listOfNumbers.insert(3,21) #insert function takes two arguments first index position at which element has to be inserted c
print(listOfNumbers)
```

```
[1, 1, 2, 21, 3, 4, 5, 15, 21, 21, 21, 21]
```

In [46]:
```
listOfNumbers=[11,6,1,2,3,4,5,1,15]
print(listOfNumbers)
print(listOfNumbers.sort()) #sort function in list doesn't return anything thats why we are getting none
print(listOfNumbers)
```

```
[11, 6, 1, 2, 3, 4, 5, 1, 15]
None
[1, 1, 2, 3, 4, 5, 6, 11, 15]
```

In [49]:
```
listOfNumbers=[11,6,-1,2,3,4,5,1,15]
print(listOfNumbers)
listOfNumbers.sort() #Natural order sorting
print(listOfNumbers)
listOfNumbers.reverse()
print(listOfNumbers)
```

```
[11, 6, -1, 2, 3, 4, 5, 1, 15]
[-1, 1, 2, 3, 4, 5, 6, 11, 15]
[15, 11, 6, 5, 4, 3, 2, 1, -1]
```

In [54]:
```
fruits =['Banana','Mango','Apple','Kiwi','Guava']
fruits.pop(1) #it takes index position as an argument and returns element at given index
print(fruits)
```

```
['Banana', 'Apple', 'Kiwi', 'Guava']
```

In [55]:
```
fruits.remove('Apple')
fruits
```

Out[55]:
```
['Banana', 'Kiwi', 'Guava']
```

In [64]:
```
listOfData=[1,5,6,2,3,4,7,9,11,10,12,13,14,15,8]
print(listOfData[4:9:2])
print(listOfData[-8:-1:1])
print(listOfData[-8:-1:2])
print(listOfData[-8:-3:2])
```

```
[3, 7, 11]
[9, 11, 10, 12, 13, 14, 15]
[9, 10, 13, 15]
[9, 10, 13]
```

In [66]:
```
#Nested List
nestedList=['Bob',28,'BLR',[9999111333,72899999,620000111],['Marathahalli','BTM','MG Road']]
phoneNumbers=nestedList[3]
print(phoneNumbers)
placesVisited=nestedList[4]
print(placesVisited)
```

```
[9999111333, 72899999, 620000111]
['Marathahalli', 'BTM', 'MG Road']
```

In [15]:
```
emptyList=list()
while(True):
    print("What type of Data you wanna insert?")
    print("1.Integer Number")
    print("2.String")
    print("3. List")
    print("4. List of Integer Number")
    print("5. List of String")
    print("6. List of List")
    print("7. Exit")

    choice= int(input("Enter your choice "))
    if choice == 1:
        intNumber = int(input("Enter integer number "))
        emptyList.insert(len(emptyList),intNumber)
        print(emptyList)
    elif choice == 2:
        strData = input("Enter string data ")
        emptyList.insert(len(emptyList),strData)
        print(emptyList)
    elif choice == 3:
        listData = input("Enter list data separated with white spaces  ").split()
        emptyList.insert(len(emptyList),listData)
        print(emptyList)
    else:
       break
```

```
What type of Data you wanna insert?
1.Integer Number
2.String
3. List
4. List of Integer Number
5. List of String
6. List of List
7. Exit
Enter your choice 1
Enter integer number 15
[15]
What type of Data you wanna insert?
1.Integer Number
2.String
3. List
4. List of Integer Number
5. List of String
6. List of List
7. Exit
Enter your choice 2
Enter string data Java
[15, 'Java']
What type of Data you wanna insert?
1.Integer Number
2.String
3. List
4. List of Integer Number
5. List of String
6. List of List
7. Exit
Enter your choice 3
Enter list data separated with white spaces  1 2 3 4 5 6
[15, 'Java', ['1', '2', '3', '4', '5', '6']]
What type of Data you wanna insert?
1.Integer Number
2.String
3. List
4. List of Integer Number
5. List of String
6. List of List
7. Exit
Enter your choice 7
```

In [17]:

```
empList=list()
listData = input("Enter list data ").split()
```

```
print(listData)
print(empList)#[]
empList.insert(len(empList),15)
print(empList)#[15]
empList.insert(len(empList),listData) #[15,['1,'2','4']]
print(empList)
```

```
Enter list data 1 2 4 5 11
['1', '2', '4', '5', '11']
[]
[15]
[15, ['1', '2', '4', '5', '11']]
```

In [20]:
```
#Create a tuple
myTuple = tuple()
print(type(myTuple))
print(myTuple)
myTuple[0]=15
```

```
<class 'tuple'>
()
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[20], line 5
      3 print(type(myTuple))
      4 print(myTuple)
----> 5 myTuple[0]=15

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

In [23]:
```
myTuple=('Biz Analysis','OB','Marketing','Finance')
print(myTuple)
print(myTuple[1])
print(dir(myTuple))
```

```
('Biz Analysis', 'OB', 'Marketing', 'Finance')
OB
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_s
ubclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__r
epr__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
```

In [25]:
```
#Dictionary
myDict =dict()
print(myDict)
print(type(myDict))
```

```
employeeDict={'name':'Sita','age':25,'city':'BLR','phone':9999111222}
print(employeeDict)
print(type(employeeDict))
```

```
{}
<class 'dict'>
{'name': 'Sita', 'age': 25, 'city': 'BLR', 'phone': 9999111222}
<class 'dict'>
```

In [26]:
```
print(dir(employeeDict))
```

```
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format
__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__ior__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex__', '__rep
r__', '__reversed__', '__ror__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'cop
y', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
```

In [40]:
```
print(employeeDict.items())
print("**********************")
print(employeeDict.keys())
print("**********************")
print(employeeDict.get('city'))
print("**********************")
print(employeeDict.values())
print("**********************")
print(employeeDict.update({'district':'BLR Urban'}))
print(employeeDict)
print("**********************")
print(employeeDict.update({'district':'BLR Rural'}))
print(employeeDict)
print("**********************")
print(employeeDict.update({'city':'Ramnagara'}))
print(employeeDict)
print("**********pop************")
print(employeeDict.pop('city'))
print(employeeDict)
print("**********popitem************")
print(employeeDict.popitem())
print(employeeDict)
```

```
dict_items([('name', 'Sita'), ('age', 25), ('phone', 9999111222), ('district', 'BLR Rural')])
**********
dict_keys(['name', 'age', 'phone', 'district'])
**********
None
**********
dict_values(['Sita', 25, 9999111222, 'BLR Rural'])
**********
None
{'name': 'Sita', 'age': 25, 'phone': 9999111222, 'district': 'BLR Urban'}
**********
None
{'name': 'Sita', 'age': 25, 'phone': 9999111222, 'district': 'BLR Rural'}
**********
None
{'name': 'Sita', 'age': 25, 'phone': 9999111222, 'district': 'BLR Rural', 'city': 'Ramnagara'}
**********pop**********
Ramnagara
{'name': 'Sita', 'age': 25, 'phone': 9999111222, 'district': 'BLR Rural'}
**********popitem**********
('district', 'BLR Rural')
{'name': 'Sita', 'age': 25, 'phone': 9999111222}
```

In [41]:
```
#User Defined Function
#function is defined
def addTwoNumbers(num1,num2):
    sum=num1+num2
    return sum

#call the function
print(addTwoNumbers(15,10))
```

```
25
```

In [43]:
```
def displayInfo():
    print("I am not returning anything")

#Calling
print(displayInfo())
```

```
I am not returning anything
None
```

In [ ]:
```
#Exercise
#Write a Python program to find those numbers which are divisible by 7 and multiples of 5, between 1500 and 2700 (both inc
```

```
#Write a Python function that takes a sequence of numbers and determines whether all the numbers are different from each o
#Write a Python program that creates all possible strings using the letters 'a', 'e', 'i', 'o', and 'I'. Ensure that each
```