In [ ]:
```
'''
LAB ASSIGNMENTS  XII - COMPUTER SCIENCE 2020-21
As per the Revised Syllabus :
Minimum  20  Python  programs.  Out  of this  at  least  4  programs
should    send    SQL commands to a database and retrieve the result.
'''
```

In [ ]:
```
"""
About Python

Python language is named after the BBC show “Monty Python’s Flying Circus”
and has nothing to do with reptiles.Since the best way to learn a language
is to use it, the video invites you to play with the Python Interpreter
as you watch.

Python was created in the early 1990s by Guido van Rossum as a successor
of a language called ABC.In 2001, the Python Software Foundation was formed,
a non-profit organization created specifically to own Python-related Intellectual
Property.

"""
```

In [81]:
```
## Operators and Operands
"""Operators are special tokens that represent computations like addition, multipli
cation and division.
The values the operator works on are called operands.
The following are all legal Python expressions whose meaning is more or less clear:
20 + 32             # 20 and 32 are operants
hour - 1            # hours and 1 are operants
hour * 60 + minute # hours , 60  and minutes are operants
minute / 60         # minutes and 60 are operants
5 ** 2              # 5 and 2 are operants
-3                  # 3 is an operand
(5 + 9) * (15 - 7)
"""
print(2 + 3)
print(2 - 3)
print(2 * 3)
print(2 ** 3)
print(3 ** 2)
minutes = 645
hours = minutes / 60
print(hours)
print(7 / 4)
print(7 // 4)
minutes = 645
hours = minutes // 60
print(hours)
quotient = 7 // 3 # This is the integer division operator
print(quotient)
remainder = 7 % 3
print(remainder)
total_secs = 7684
hours = total_secs // 3600
```

```
secs_still_remaining = total_secs % 3600
minutes =  secs_still_remaining // 60
secs_finally_remaining = secs_still_remaining  % 60
print(hours)
print(secs_still_remaining)
print(minutes)
print(secs_finally_remaining)
print(2 ** 3 ** 2)      # the right-most ** operator gets done first!
print((2 ** 3) ** 2)    # use parentheses to force the order you want!
```

```
5
-1
6
8
9
10.75
1.75
1
10
2
1
2
484
8
4
512
64
```

In [ ]:
```
#Strings
#What is a String?
#Strings are sets of characters.
#In Python strings are in triple or double or single quotes like
"Hello World!" or 'Hello World!' or ''''Hello World'''
#A commonly used string in our day to day life is our name.
```

In [85]:
```
#Character Example : A, B, C, a, b, c, 0,1, 2, 3, @ ,%, & ,[ ,]
'My India' # single quotes
"My India" # double quotes
'''My India''' # Triple quotes
'doesn\'t' # use \' to escape the single quote.
"doesn't" # or use double quotes instead
'"Yes," they said.' # or use single quotes instead
"\"Yes,\" they said." # use \" to escape the double quote.
```

Out[85]:
```
'"Yes," they said.'
```

In [84]:
```
# What is \n

"""
\n is a special character sequence. This sequences are called escape
characters and are there to denote special meaning to a character in
a string.The '\n' sequence is a popular one found in many languages
that support escape sequences.It is used to indicate a new line in a
string.

"""
```

Out[84]:
```
"\n\n is a special character sequence. This sequences are called escape\ncharact
ers and are there to denote special meaning to a character in\na string.The '\n'
sequence is a popular one found in many languages\nthat support escape sequence
s.It is used to indicate a new line in a\nstring.\n\n"
```

In [86]:
```
# Use of \n without print()

s="First line\nSecond line"
s   # without print(), \n is included in the output
```

Out[86]:
```
'First line\nSecond line'
```

In [87]:
```
# Use of \n with print()

s="First line\nSecond line"
print(s)   # with print(), \n produces a new line
```

```
First line
Second line
```

In [88]:
```
"""
If you don't want characters prefaced by \ to be interpreted as
special characters,you can use raw strings by adding an r before
the first quote:

"""
#print without raw string

print('seco\nd') # here \n means newline!
```

```
seco
d
```

In [89]:
```
"""
If you don't want characters prefaced by \ to be interpreted as
special characters,you can use raw strings by adding an r before
the first quote:

"""
#print with raw string using r

print(r'seco\nd') # note the r before the quote . It print \ with string
```

```
seco\nd
```

In [90]:
```
'''
String literals can span multiple lines. One way is using triple-quotes:
End of lines are automatically included in the string, but it's possible
to prevent this by adding a \ at the end of the line.
'''
```

Out[90]:
```
'\nString literals can span multiple lines. One way is using triple-quotes:\nEnd
of lines are automatically included in the string, but it's possible\nto prevent
this by adding a \\ at the end of the line.\n'
```

In [91]:
```
# Use of \ at the end of the line
print("""\
Choose Your Option:\
1. New Customer\
2. Display Customer\
""")
```

```
Choose Your Option:1. New Customer2. Display Customer
```

In [92]:
```
# Without using \ at the end of the line
print("""
Choose Your Option:
1. New Customer
2. Display Customer
""")
```

```
Choose Your Option:
1. New Customer
2. Display Customer
```

In [93]:
```
# Use of * and + operators with Strings
"""
Strings can be concatenated (glued together) with the
+ operator, and repeated with *:

"""

3 * 'w' + 'e'    # 3 times 'w', followed by 'e'
```

Out[93]: 'wwwe'

In [94]:
```
"""
Two or more string literals (i.e. the ones enclosed between quotes)
next to each other are automatically concatenated.

"""

'Aao' 'Milke' 'Chale'
```

Out[94]: 'AaoMilkeChale'

In [95]:
```
"""
Two or more string literals (i.e. the ones enclosed between quotes
[ different types] ) next to each other are automatically concatenated.

"""

"Aao" 'Milke' 'Chale'
```

Out[95]: 'AaoMilkeChale'

In [96]:
```
#This feature is particularly useful when you want to break long strings:
text =('Put several strings within parentheses '
       'to have them joined together.')
text
```

Out[96]:
```
'Put several strings within parentheses to have them joined together.'
```

In [97]:
```
# This only works with two literals though, not with variables or expressions:
prefix = 'Py'
prefix 'thon' # can't concatenate a variable and a string literal
```

```
  File "<ipython-input-97-b504f0f3d7d5>", line 3
    prefix 'thon' # can't concatenate a variable and a string literal
                  ^
SyntaxError: invalid syntax
```

In [98]:
```
#If you want to concatenate variables or a variable and a literal, use +:
prefix = 'Py'
prefix + 'thon' # can concatenate a variable and a string literal
```

Out[98]:
```
'Python'
```

In [100]:
```
"""
Strings can be indexed (subscripted), with the first character having index 0.
There is no separate character type; a character is simply a string of size one:
"""
word = 'Python'

"""
One way to remember how strings are indexed internally we can make a chart
like this:
+---+---+---+---+---+---+
| P | y | t | h | o | n |
+---+---+---+---+---+---+
  0   1   2   3   4   5
 -6  -5  -4  -3  -2  -1

Note that since -0 is the same as 0, negative indices start from -1.

"""

word[0] # character in position 0
word[5] # character in position 5
#Indices may also be negative numbers, to start counting from the right:
word[-1] # last character
word[-2] # second-last character
word[-6] # first character
#Slicing

"""
In addition to indexing, slicing is also supported.
While indexing is used to obtain individual characters,
slicing allows you to obtain substring:
"""
```

```
word = 'Python'
word[0:2] # characters from position 0 (included) to 2 (excluded)
word[2:5] # characters from position 2 (included) to 5 (excluded)
"""
Note how the start is always included, and the end always excluded.
This makes sure that s[:i] + s[i:] is always equal to s:

word[0:2] + word[2:0] = Full String

"""
word="Python"
print(word[:2] + word[2:])
"""
Slice indices have useful defaults;
an omitted first index defaults to zero, an omitted second index defaults
to the size of the string being sliced.
"""
word[:2] # character from the beginning to position 2 (excluded)
word[4:] # characters from position 4 (included) to the end
word[-2:] # characters from the second-last (included) to the end
#Attempting to use an index that is too large will result in an error:
#word[10]
"""
However, out of range slice indexes are handled gracefully when used
for slicing.Attempting to use a starting index that is larger than the
string internal index will result an empty string:

"""
word[6:] # Max postive index is 5
"""
Attempting to use a ending index that is larger than the string internal
index will result an orginal string:
```

```
"""
word[:6] # Max postive index is 5
"""
Attempting to use a starting index larger than ending index
will result an empty string:

"""
word[3:0] # Max postive index is 5
"""
Working with negative index:

+---+---+---+---+---+---+
| P | y | t | h | o | n |
+---+---+---+---+---+---+
  0   1   2   3   4   5
 -6  -5  -4  -3  -2  -1

"""
word ="Python"
print(word[0:-3]) # 'Pyt'

"""
Working with negative index:

+---+---+---+---+---+---+
| P | y | t | h | o | n |
+---+---+---+---+---+---+
  0   1   2   3   4   5
 -6  -5  -4  -3  -2  -1

No output as slicing only works with left to right
```

```
and the ending index must be to the left postion of starting index
or at the same position

"""
word[-3:-6] # No output
# In Slicing don't overlap starting index with ending index.
```

```
Python
Pyt
```

Out[100]:
```
''
```

In [83]:
```
#Quotes Must Matched

'My India" # Quotes Mis-Matched
```

```
  File "<ipython-input-83-35f1363fc770>", line 3
    'My India" # Quotes Mis-Matched
                                   ^
SyntaxError: EOL while scanning string literal
```

In [74]:
```
# use of math module in Python
import math
print(math.sqrt(9))
print(math.pow(2,3))
```

```
3.0
8.0
```

In [67]:
```
# Python program to implement exponential Operator
num1 = 4
num2 = 3
num3 = 2
num1 = num1 ** (num2 + num3)
print (num1)
```

```
1024
```

In [69]:
```
#Creating and accessing Dictionary keys
d ={ 0: 'a' , 1 : 'b' , 2:'c' ,3:'d' }
for i in d:
    print(i)
```

```
0
1
2
3
```

In [70]:
```
#Creating and accessing Dictionary values
d ={ 0: 'a' , 1 : 'b' , 2:'c' ,3:'d' }
for i in d:
    print(d[i])
```

```
a
b
c
d
```

In [72]:
```
# Comparing Two Dictionaries
d1 = {"john":40, "peter":45}
d2 = {"john":466, "peter":45}
d1 == d2
```

Out[72]:
```
False
```

In [71]:
```
#Creating and accessing Dictionary keys and values
d ={ 0: 'a' , 1 : 'b' , 2:'c' ,3:'d' }
for i in d:
    print("Key :=",i,"Value :=",d[i])
```

```
Key := 0 Value := a
Key := 1 Value := b
Key := 2 Value := c
Key := 3 Value := d
```

In [64]:
```
# Dictionary Implementation
myDict = {'three': 'tres', 'one': 'uno', 'two': 'dos'}
value = myDict['two']
print(value)
print(myDict['two'])
myDict['four']='Firefox'
print(myDict)
myDict['three'] = 'Google'
print(myDict)
myDict['thre'] = 'IOS'
print(myDict)
myDict[5] = 'New Integer key'
print(myDict)
for key in myDict:
    print(myDict[key])
print(myDict.keys())
print(myDict.values())
```

```
dos
dos
{'three': 'tres', 'one': 'uno', 'two': 'dos', 'four': 'Firefox'}
{'three': 'Google', 'one': 'uno', 'two': 'dos', 'four': 'Firefox'}
{'three': 'Google', 'one': 'uno', 'two': 'dos', 'four': 'Firefox', 'thre': 'IOS
'}
{'three': 'Google', 'one': 'uno', 'two': 'dos', 'four': 'Firefox', 'thre': 'IOS
', 5: 'New Integer key'}
Google
uno
dos
Firefox
IOS
New Integer key
dict_keys(['three', 'one', 'two', 'four', 'thre', 5])
dict_values(['Google', 'uno', 'dos', 'Firefox', 'IOS', 'New Integer key'])
```

In [65]:
```
# Passing and Modifying Dictionary in a Function
def passDict(myDict):
    for key in myDict:
        print(myDict[key])
def modifyDict(myDict):
     myDict[4] = 'Four'

myDict = {1:'One',2 : 'Two' ,3:'Three'}
passDict(myDict)
print(myDict)
modifyDict(myDict)
print(myDict)
```

```
One
Two
Three
{1: 'One', 2: 'Two', 3: 'Three'}
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four'}
```

In [60]:
```
# Passing Tuple in a Function
def passList(myTuple):
    for position in range(len(myTuple)):
        print(myTuple[position])

myTuple = [2, 5, 9]
passList(myTuple)
```

```
2
5
9
```

In [61]:
```
# Passing Tuple in a Function entering tuple elements directly within function call

def passList(myTuple):
    for position in range(len(myTuple)):
        print(myTuple[position])

passList((2, 5, 9))
```

```
2
5
9
```

In [62]:
```
#Passing Tuple with mixed data in a Function

def passList(myTuple):
    for position in range(len(myTuple)):
        print(myTuple[position])

passList(('String 1','String 2',10, 10.5 , True))
```

```
String 1
String 2
10
10.5
True
```

In [63]:
```
#We cannot Modify Tuple Inside Function
print("These code will generate an error as Tuples are immutable")

def doubleStuff(myTuple):
    """ Try to Overwrite each element in myTuple with double its value. """
    for position in range(len(myTuple)):
        myTuple[position] = 2 * myTuple[position]

myTuple = (2, 5, 9)
print(myTuple)
doubleStuff(myTuple)
```

```
These code will generate an error as Tuples are immutable
(2, 5, 9)

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-63-4bff4907304c> in <module>()
      9 myTuple = (2, 5, 9)
     10 print(myTuple)
---> 11 doubleStuff(myTuple)

<ipython-input-63-4bff4907304c> in doubleStuff(myTuple)
      5     """ Try to Overwrite each element in myTuple with double its value.
"""
      6     for position in range(len(myTuple)):
----> 7         myTuple[position] = 2 * myTuple[position]
      8
      9 myTuple = (2, 5, 9)

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

In [58]:
```
# Passing List with mixed data in a Function
def passList(myList):
    for position in range(len(myList)):
        print(myList[position])
passList(['String 1','String 2',10, 10.5 , True])
```

```
String 1
String 2
10
10.5
True
```

In [57]:
```
# Passing List in a Function entering list directly within function call
def passList(myList):
    for position in range(len(myList)):
        print(myList[position])
passList([2, 5, 9])
```

```
2
5
9
```

In [56]:
```
# Passing List in a Function
def passList(myList):
    for position in range(len(myList)):
        print(myList[position])

myList = [2, 5, 9]
passList(myList)
```

```
2
5
9
```

In [59]:
```
#Modifying List Inside Function

print("""
 Lists are mutable.Functions which take lists as arguments and change them during e
xecution
 are called modifiers and the changes they make are called side effects.
 Passing a list as an argument actually passes a reference to the list,
 not a copy of the list. Since lists are mutable, changes made to the elements
 referenced by the parameter change the same list that the argument is referencing.
 For example, the function below takes a list as an argument and multiplies
 each element in the list by 2:
""")

def doubleStuff(aList):
    """ Overwrite each element in aList with double its value. """
    for position in range(len(aList)):
        aList[position] = 2 * aList[position]

things = [2, 5, 9]
print(things)
doubleStuff(things)

print("Äfter function call the list will be modified\n",things)

print("""
The parameter aList and the variable things are aliases for the same object.
Since the list object is shared by two references, there is only one copy.
If a function modifies the elements of a list parameter, the caller sees the
change since the change is occurring to the original.Note that after the call
to doubleStuff, the formal parameter aList refers to the same object as the
actual parameter things. There is only one copy of the list object itself.
""")
```

```
 Lists are mutable.Functions which take lists as arguments and change them durin
g execution
 are called modifiers and the changes they make are called side effects.
 Passing a list as an argument actually passes a reference to the list,
 not a copy of the list. Since lists are mutable, changes made to the elements
 referenced by the parameter change the same list that the argument is referenci
ng.
 For example, the function below takes a list as an argument and multiplies
 each element in the list by 2:

[2, 5, 9]
Äfter function call the list will be modified
 [4, 10, 18]

The parameter aList and the variable things are aliases for the same object.
Since the list object is shared by two references, there is only one copy.
If a function modifies the elements of a list parameter, the caller sees the
change since the change is occurring to the original.Note that after the call
to doubleStuff, the formal parameter aList refers to the same object as the
actual parameter things. There is only one copy of the list object itself.
```

In [55]:
```
def addnum():
    fnum = int(input("Enter first number: "))
    snum = int(input("Enter second number: "))
    sum = fnum + snum
    print("The sum of ",fnum,"and ",snum,"is ",sum)

#function call
addnum()
```

```
Enter first number: 45
Enter second number: 67
The sum of  45 and  67 is  112
```

In [54]:
```
num = 5 #Global variable
def myfunc1():
 #Prefixing global informs Python to use the updated global
 #variable num outside the function
     global num
     num = 10
     temp=20 # local variable
     print("num reassigned =",num)
#function ends here
myfunc1()
print("Accessing num outside myfunc1",num)
print(temp)
```

```
num reassigned = 10
Accessing num outside myfunc1 10

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-54-13412cc8de92> in <module>()
     10 myfunc1()
     11 print("Accessing num outside myfunc1",num)
---> 12 print(temp)

NameError: name 'temp' is not defined
```

In [53]:
```
#Check whether a Number is Armstrong or Not
print("An Armstrong number is an n-digit number that is equal to the sum of the nth powers of its digits.")
def is_armstrong(x):
    t=x

    #Codes to find the Number of Digits in the Number
    ndigit=0
    while t >0:
        ndigit =ndigit +1
        t=t//10

    #Code to Check Armstrong Number

    n=x
    s=0
    while x >0:
        m= x %10
        s=s+m**ndigit
        x=x//10
    if s==n:
        return "The Number is ArmStrong - SUM = ",s,"Number = ",n
    else:
        return "The Number is Not ArmStrong - SUM = ",s,"Number = ",n

num=int(input("Enter a Number "))
is_armstrong(num)
```

```
An Armstrong number is an n-digit number that is equal to the sum of the nth pow
ers of its digits.
Enter a Number 175
```

Out[53]:
```
('The Number is Not ArmStrong - SUM = ', 469, 'Number = ', 175)
```

In [66]:
```
# How to use two return statements in a function
def SumOfTwoNumbers(x,y):
    z= x+y
    w= x*y
    return z,w # Returning more than one value

a= int(input("Enter First Number"))
b= int(input("Enter First Number"))
s,t=SumOfTwoNumbers(a,b)
print("The Sum is :" ,s)
print("The Productis :" ,t)
```

```
Enter First Number45
Enter First Number78
The Sum is : 123
The Productis : 3510
```

In [73]:
```
#Common Functions in Python
print(str(12.5))
print(float(12))
print(int('12'))
n=input("Enter a Number")
print(n)
print('45+10')
print(45+10)
print(eval('45+10'))
exp = input("Enter a valid Mathematical expression")
print(exp)
res =eval(exp)
print(res)
print(max([32,56,43,12,89]))
print(max(45,32,56))
print(max('Ajay','AjaY','Seema','Bony'))
print(min(45,32,56))
print(min('Ajay','AjaY','Seema','Bony'))
print(abs(-5))
print(len([1,2,3,4,5]))
print(len('Palampur'))
for i in range(1,10,2):
    print(i,end=' ')
print() # Blank Line
for i in range(1,10,2):
    print(i,end='@')
```

```
12.5
12.0
12
Enter a Number45
45
45+10
55
55
Enter a valid Mathematical expression2+3
2+3
5
89
56
Seema
32
AjaY
5
5
8
1 3 5 7 9
1@3@5@7@9@
```

In [51]:
```
# Stack Implementation using List
mystack = []
while True:
    num= int(input("Enter Element of the Stack [ Enter 0 to End]"))
    if num == 0:
        break
    else:
        mystack.append(num)
print("POP Operation from the Stack")
while mystack:
    print(mystack.pop())
```

```
Enter Element of the Stack [ Enter 0 to End]45
Enter Element of the Stack [ Enter 0 to End]43
Enter Element of the Stack [ Enter 0 to End]67
Enter Element of the Stack [ Enter 0 to End]54
Enter Element of the Stack [ Enter 0 to End]0
POP Operation from the Stack
54
67
43
45
```

In [52]:
```
# Queue Implementation using List
myqueue = []
while True:
    num= int(input("Enter Element of the Queue [ Enter 0 to End]"))
    if num == 0:
        break
    else:
        myqueue.append(num)
print("Delete Operation from the Queue")
i=0
while i< len(myqueue):
   print(myqueue[i])
   i=i+1
```

```
Enter Element of the Queue [ Enter 0 to End]78
Enter Element of the Queue [ Enter 0 to End]65
Enter Element of the Queue [ Enter 0 to End]78
Enter Element of the Queue [ Enter 0 to End]56
Enter Element of the Queue [ Enter 0 to End]0
Delete Operation from the Queue
78
65
78
56
```

In [50]:
```
# SORTING OF LIST IN DESCENDING ORDER
n= int(input("Enter How many Numbers"))
mylist = []
i=0
while i<n:
    num= int(input("Enter The Element of the List"))
    mylist.append(num)
    i=i+1

print("Original List is")
print(mylist)

mylist.sort(reverse=True) # Sorting of List
print("List after sorting in Descending order")
print(mylist)
```

```
Enter How many Numbers5
Enter The Element of the List45
Enter The Element of the List65
Enter The Element of the List34
Enter The Element of the List23
Enter The Element of the List67
Original List is
[45, 65, 34, 23, 67]
List after sorting in Descending order
[67, 65, 45, 34, 23]
```

In [49]:
```
# Use of Dictionary argument in Function

def texample(two):
    two['Name'] ="Karan"  # Change of Name key value
    print("Inside texample function\n",two)

two={'Name' : 'Radha' , 'Age':30}
print("Initial Dictionary is ",two)
texample(two)
print("Outside texample function - Change reflected \n",two)
```

```
Initial Dictionary is  {'Name': 'Radha', 'Age': 30}
Inside texample function
 {'Name': 'Karan', 'Age': 30}
Outside texample function - Change reflected
 {'Name': 'Karan', 'Age': 30}
```

In [48]:
```
# Use of Dictionary argument in Function

def texample(**two):
    print("Inside texample function\n",two)

two={'Name' : 'Radha' , 'Age':30}
print("Initial Dictionary is ",two)
texample(**two)
print("Outside texample function \n",two)
```

```
Initial Dictionary is  {'Name': 'Radha', 'Age': 30}
Inside texample function
 {'Name': 'Radha', 'Age': 30}
Outside texample function
 {'Name': 'Radha', 'Age': 30}
```

In [47]:
```
# Use of List Argument as a Tuple in Function
def texample(*two):
    print("\nInside texample Function\n")
    for i in two:
     print(i ,"\t")

list=[1,2,3,4]
print("\nCalled with texample(list) - As a Single Value\n")
texample(list) # Passing list as a single argument
print("\nCalled with texample(*tlist) - As a Tuple of Values \n")
texample(*list) # Passing list  as a tuple argument
##print("\nCalled with texample([1,2,3,4])\n")
```

```
Called with texample(list) - As a Single Value


Inside texample Function

[1, 2, 3, 4]

Called with texample(*tlist) - As a Tuple of Values


Inside texample Function

1
2
3
4
```

In [46]:
```
# Use of Tuple Argument and List Arguments as a Tuple in Function
def texample(*two):
    for i in two:
     print(i ,"\t")
t=(1,2,3) # Tuple
list=[1,2,3,4]
print("\nCalled with texample(t)\n")
texample(t) # Passing tuple as a single argument
print("\nCalled with texample(*t)\n")
texample(*t) # Passing tuple as a variable argument
print("\nCalled with texample([1,2,3,4])\n")
texample(list) # Passing list as single argument
print("\nCalled with texample(*t,[1,2,3,4])\n")
texample(*t,list) # Passing tuple as variable argument and list as a single argumen
t
print("\nCalled with texample(t,[1,2,3,4])\n")
texample(t,list) # Passing tuple and list as a single argument
```

```
Called with texample(t)

(1, 2, 3)

Called with texample(*t)

1
2
3

Called with texample([1,2,3,4])

[1, 2, 3, 4]

Called with texample(*t,[1,2,3,4])

1
2
3
[1, 2, 3, 4]

Called with texample(t,[1,2,3,4])

(1, 2, 3)
[1, 2, 3, 4]
```

In [45]:
```
# Use of Tuple Arguments in Function
def calculateSum(*args):
    sumOfNumbers = 0
    for elem in args:
        sumOfNumbers += elem
    return sumOfNumbers

tuple = (1,2,3,4,5)
sum = calculateSum( *tuple)
print("\nSum of List Elements is = " , sum)
```

```
Sum of List Elements is =  15
```

In [44]:
```
# Use of and tuple Arguments in Function
def texample(two):
    two=(5,6,7,8)    # Updating the Tuple
    print("\nInside texample Function\n")
    for i in two:
     print(i ,"\t")
tuple=(1,2,3,4)
print("\nOrginal Tuple \n",tuple)
print("\nCalled with texample(tuple)\n")
texample(tuple) # Passing tuple as a argument
print("\nNo Updation Reflected \n",tuple)
```

```
Orginal Tuple
 (1, 2, 3, 4)

Called with texample(tuple)


Inside texample Function

5
6
7
8

No Updation Reflected
 (1, 2, 3, 4)
```

In [43]:
```
# Use of List Arguments in Function
def calculateSum(*args):
    sumOfNumbers = 0
    for elem in args:
        sumOfNumbers += elem
    return sumOfNumbers

list1 = [1,2,3,4,5,6,7,8]
sum = calculateSum( *list1)
print("\nSum of List Elements is = " , sum)
```

```
Sum of List Elements is =  36
```

In [42]:
```
# Use of List Arguments in Function
def texample(two):
    two[0]=100   # Updating the Value of List
    print("\nInside texample Function\n")
    for i in two:
     print(i ,"\t")

list=[1,2,3,4]
print("\nOrginal List \n",list)
print("\nCalled with texample(list)\n")
texample(list) # Passing list as argument
print("\nUpdated List \n",list)
```

```
Orginal List
 [1, 2, 3, 4]

Called with texample(list)


Inside texample Function

100
2
3
4

Updated List
 [100, 2, 3, 4]
```

In [41]:
```
# Use of One Default and One actual Argument
def Dfunction(x,y=20): # Default argument must follows non-default argument

   print("\n x = " , x , "y = ",y)

print("\ncalled with Dfunction(10) ")
Dfunction(10) # Called with Single Argument i.e x
print("\ncalled with Dfunction(10,200) ")
Dfunction(10,200)
```

```
called with Dfunction(10)

 x =  10 y =  20

called with Dfunction(10,200)

 x =  10 y =  200
```

In [40]:
```
# Use of Default Argument
def reverseOfNumber(num=789): # Definition of  Function and Default Argument
    r=0

    while(num > 0) :
            m= num%10
            r=r * 10 + m
            num= num//10
    return r
number= int(input("\nEnter a Number\t"))

reverse = reverseOfNumber() # Call to Function with No argument
print("\n The function will take its Default Argument 789 and calculate the revers
e")
reverse = reverseOfNumber() # Call to Function with No argument
print("\n The Reverse of the Number is " , reverse)

print("\n The function will take its Argument as entered by the user and calculate
the reverse")
reverse = reverseOfNumber(number) # Call to Function with With argument
print("\n The Reverse of the Number is " , reverse)
```

```
Enter a Number  34

 The function will take its Default Argument 789 and calculate the reverse

 The Reverse of the Number is  987

 The function will take its Argument as entered by the user and calculate the re
verse

 The Reverse of the Number is  43
```

In [38]:
```
# Use of Return statement in Function
def texample(x ,y):
     return x+y
x= int(input("\n Input the value of x"))
y= int(input("\n Input the value of y"))
print("\nOutside texample Function")
print("\n x = ",x, "\t y = ",y)
s =texample(x,y)
print("\n Sum is ",texample(x,y))
```

```
 Input the value of x3

 Input the value of y4

Outside texample Function

 x =  3          y =  4

 Sum is  7
```

In [39]:
```
# Use of Function Argument and Return Statement
def reverseOfNumber(num): # Definition of  Function
    r=0
    while(num > 0) :
            m= num%10
            r=r * 10 + m
            num= num//10
    return r
number= int(input("\nEnter a Number\t"))
reverse = reverseOfNumber(number) # Call to Function with argument
print("\n The Reverse of the Number is " , reverse) # Call to Function with argumen
t
```

```
Enter a Number  234

 The Reverse of the Number is  432
```

In [37]:
```
# Use of Function Argument
def reverseOfNumber(num): # Definition of  Function
    reverse=0
    while(num > 0) :
            m= num%10
            reverse=reverse * 10 + m
            num= num//10
    print("\n The Reverse of the Number is " , reverse)

number= int(input("\nEnter a Number\t"))
reverseOfNumber(number) # Call to Function with argument
```

```
Enter a Number  123

 The Reverse of the Number is  321
```

In [36]:
```
# Use of Parameters/Arguments in Function
def texample(x,y): # formal parameters/arguments
    x=40
    y=50
    print("\nInside texample Function")
    print("\n x = ",x, "\t y = ",y)

x= int(input("\n Input the value of x"))
y= int(input("\n Input the value of y"))
texample(x,y)# Actual parameters/arguments
print("\nOutside texample Function")
print("\n x = ",x, "\t y = ",y)
```

```
 Input the value of x2

 Input the value of y3

Inside texample Function

 x =  40         y =  50

Outside texample Function

 x =  2          y =  3
```

In [34]:
```
# Use of Function
def sumOfDigits(): # Definition of  Function
    s=0
    n = int(input("\nEnter a Number\t"))
    while(n > 0) :
            m= n%10
            s=s+m
            n= n//10
    print("\n The Sum of Digits of the above number is   ", s )

sumOfDigits() # Call to Function
```

```
Enter a Number  123

 The Sum of Digits of the above number is    6
```

In [3]:
```
#  Read a text file line by line and display each word separated by a #.
f=open("abc.txt","r")
fullContents=f.read() # This method will creates a string of all lines.
words =fullContents.split()# This method will creates a list of words.

for w in words:  # Read each word
    print(w,end='#')
f.close()
```

```
New#Appended#contentsOne#more#lineNew#data#has#been#Appended#
```

In [33]:
```
# Remove all the lines that contain the character `a' in a file and write it to ano
ther file.
flag =0
f=open("abc.txt","r")
destination= open("dest.txt" ,"w")
data=f.readlines()
print("Initial File Contents")
print(data)
for line in data:
    flag=0
    for ch in line:
            if ch == 'a':
                flag=1
                break
    if flag==0:
        destination.write(line)

f.close()
destination.close()
destination=open("dest.txt","r")
data=destination.read()
print("\nAll lines that do not contain the character `a' are written to a new fil
e.")
print(data)
destination.close()
```

```
Initial File Contents
['New Appended contentsOne more lineNew data has been Appended\n', 'A new method
to write Python program\n', 'Only This Line Left']

All lines that do not contain the character `a' are written to a new file.
Only This Line Left
```

In [21]:
```
'''Read  a  text  file  and  display  the  number  of
vowels/  consonants/ uppercase/ lowercase characters in the file. '''
f=open("abc.txt","r")
fullContents=f.read() # This method will creates a string of all lines.
vowels=0
upper=0
lower=0
consonents=0
digits=0
vowelsString ='AEIOUaeiou'
consonentsString ='bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQR'
upperAlphabets='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
lowerAlphabets='abcdefghijklmnopqrstuvwxyz'
digitsString='0123456789'
print("File Contents Are:")
print(fullContents)
for ch in fullContents:
    if ch in vowelsString:
        vowels = vowels +1
    elif ch in consonentsString:
        consonents =consonents +1
    elif ch in digitsString:
        digits = digits +1
    if ch in upperAlphabets:
        upper=upper+1
    elif ch in lowerAlphabets:
        lower=lower+1
print("Vowels =:",vowels)
print("Uppercase =",upper)
print("Lowercase =",lower)
print("Consonents",consonents)
print("Digits",digits)
```

```
f.close()
```

```
File Contents Are:
New Appended contentsOne more lineNew data has been Appended
Vowels =: 21
Uppercase = 5
Lowercase = 47
Consonents 31
Digits 0
```

In [1]:
```
# Random number generator that generates random numbers between 1 and 6 (simulates
a dice).
import random
ans ='y'
print("Roll Dice")
while ans=='y' or ans == 'Y':
    number=random.randint(1,6)
    print("The number is :",number)
    ans=input("Roll Again : Please press y or Y ")
```

```
Roll Dice
The number is : 4
Roll Again : Please press y or Y y
The number is : 5
Roll Again : Please press y or Y y
The number is : 2
Roll Again : Please press y or Y Y
The number is : 3
Roll Again : Please press y or Y n
```

In [16]:
```
# SATACK - LAST IN FIRST OUT
# Python Implementation of Stack using List

def DisplayStack(MyStack):
       print(MyStack)

def Push(Value):
    if len(MyStack) < StackSize:
        MyStack.append(Value)
    else:
        print("Stack is full!")

def Pop():
    if len(MyStack) > 0:
        MyStack.pop()
    else:
        print("Stack is empty.")

MyStack = [] # Empty Stack
StackSize = 3 # Size of Stack after three Push Operations

print("3 Push Operations in the Stack ")


Push(1)
print("Stack after Push Operation Contains:")
DisplayStack(MyStack)

Push(2)
print("Stack after Push Operation Contains:")
DisplayStack(MyStack)
```

```
Push(3)
print("Stack after Push Operation Contains:")
DisplayStack(MyStack)

Push(4)

print("\n\n3 Pop Operations from the Stack ")
DisplayStack(MyStack)

Pop()
print("Stack after Pop Operation Contains:")
DisplayStack(MyStack)

Pop()
print("Stack after Pop Operation Contains:")
DisplayStack(MyStack)

Pop()
print("Stack after Pop Operation Contains:")
DisplayStack(MyStack)

Pop()
```

```
3 Push Operations in the Stack
Stack after Push Operation Contains:
[1]
Stack after Push Operation Contains:
[1, 2]
Stack after Push Operation Contains:
[1, 2, 3]
Stack is full!


3 Pop Operations from the Stack
[1, 2, 3]
Stack after Pop Operation Contains:
[1, 2]
Stack after Pop Operation Contains:
[1]
Stack after Pop Operation Contains:
[]
Stack is empty.
```

In [17]:
```
# QUEUE - FIRST IN FIRST OUT
# Python Implementation of Queue Using List

def DisplayQueue(MyQueue):
       print(MyQueue)

def EnQueue(Value):
    if len(MyQueue) < QueueSize:
        MyQueue.append(Value)
    else:
        print("Queue is full!")

def DeQueue():
    if len(MyQueue) > 0:
        del(MyQueue[0])
    else:
        print("Queue is empty.")

MyQueue = [] # Empty Stack
QueueSize = 3 # Size of Stack after three Push Operations

print("3 EnQueue or Insert Operations in the Queue ")


EnQueue(1)
print("Queue after EnQueue Operation Contains:")
DisplayQueue(MyQueue)

EnQueue(2)
print("Queue after EnQueue Operation Contains:")
DisplayQueue(MyQueue)
```

```
EnQueue(3)
print("Queue after EnQueue Operation Contains:")
DisplayQueue(MyQueue)

EnQueue(4)

print("\n\n3 DeQueue or Delete Operations from the Queue ")
DisplayQueue(MyQueue)

DeQueue()
print("Queue after DeQueue Operation Contains:")
DisplayQueue(MyQueue)

DeQueue()
print("Queue after DeQueue Operation Contains:")
DisplayQueue(MyQueue)

DeQueue()
print("Queue after DeQueue Operation Contains:")
DisplayQueue(MyQueue)

DeQueue()
```

```
3 EnQueue or Insert Operations in the Queue
Queue after EnQueue Operation Contains:
[1]
Queue after EnQueue Operation Contains:
[1, 2]
Queue after EnQueue Operation Contains:
[1, 2, 3]
Queue is full!


3 DeQueue or Delete Operations from the Queue
[1, 2, 3]
Queue after DeQueue Operation Contains:
[2, 3]
Queue after DeQueue Operation Contains:
[3]
Queue after DeQueue Operation Contains:
[]
Queue is empty.
```

In [25]:
```
# SORTING OF LIST IN DESCENDING ORDER
n= int(input("Enter How many Numbers"))
mylist = []
i=0
while i<n:
    num= int(input("Enter The Element of the List"))
    mylist.append(num)
    i=i+1

print("Original List is")
print(mylist)

mylist.sort(reverse=True) # Sorting of List
print("List after sorting in Descending order")
print(mylist)
```

```
Enter How many Numbers5
Enter The Element of the List3
Enter The Element of the List4
Enter The Element of the List25
Enter The Element of the List12
Enter The Element of the List89
Original List is
[3, 4, 25, 12, 89]
List after sorting in Descending order
[89, 25, 12, 4, 3]
```

In [25]:
```
#  A PYTHON PROGRAM  TO FIND THE NUMBER OF WORDS IN A TEXT FILE
f=open("abc.txt","r")
linesList=f.readlines() # This method will creates a List of all lines
count=0
for line in linesList:  # Read each line
    wordsList=line.split() # THIS FUNCTION IS USED TO BREAK THE LINE INTO LIST OF W
ORDS
    print(wordsList) # No Need to Print
    count = count+ len(wordsList)
print("The number of words in this file are  : ",count)
f.close()
```

```
['New', 'Appended', 'contentsOne', 'more', 'lineNew', 'data', 'has', 'been', 'Ap
pended']
The number of words in this file are  :  9
```

In [29]:
```
#  A PYTHON PROGRAM TO COUNT ALL THE LINE HAVING 'a/A' AS FIRST CHARACTER
count =0
f=open("abc.txt","r")
data=f.readlines()
print(data)
for line in data:
      if line[0] == 'A' or line[0] == 'a':
        count=count+1
print("Number of lines having 'a/A' as first character is/are : " ,count)
f.close()
```

```
['New Appended contentsOne more lineNew data has been Appended\n', 'A new method
to write Python program']
Number of lines having 'a/A' as first character is/are :  1
```

In [31]:
```
#  A PYTHON PROGRAM TO FIND THE FREQUENCY OF A WORD IN A TEXT FILE
count =0
f=open("abc.txt","r")
data=f.read()
wordsList = data.split() # THIS FUNCTION IS USED TO BREAK THE LINE INTO LIST OF WOR
DS
key = input ("\nEnter the word to find its frequnecy in the File : ")
for word in wordsList:
    if word == key:
        count =count +1
print("\n The frequency of word " , key , "  is " ,count)

f.close()
```

```
Enter the word to find its frequnecy in the File : new

 The frequency of word  new   is  1
```

In [33]:
```
#  A PYTHON PROGRAM TO CREATE A BINARY FILE

import pickle  # USE OF pickle Library

binaryfile=open("binary.dat","wb") # Open file in Binary Mode
text =input("Enter the text for Binary File : ")

pickle.dump(text , binaryfile) # USE OF pickle's dump() method

binaryfile.close()
```

```
Enter the text for Binary File : Proud to be KVIAN
```

In [ ]:
```
"""
Create a binary file with name and roll number. Search for a given
roll number  and  display  the  name,if  not  found  display
appropriate message.
"""

import pickle
f =open("bfile.dat","wb")
dict = { 'Name' : ['Laxmi','Sita','Radha'],'rollNo' : [12,14,10] }
pickle.dump(dict , f)
f.close()

f =open("bfile.dat","rb")
data=pickle.load(f)
age=int(input("Enter the Roll No You want to Search"))

n=len(data['Age'])
i=0
while i<n:    # Iterating through the File to Search the Record
    if data['rollNo'][i] == age:
        print("Record Found")
        break;
    i=i+1
f.close()

if i==n:  #This will true if not record found
  print("Record Not Found")
```

In [23]:
```
"""
Create a binary file with roll number, name and marks.
Input a roll number and update the marks.
"""

import pickle
f =open("bfile.dat","wb")
dict = { 'Name' : ['Laxmi','Sita','Radha'],'rollNo' : [12,14,10] }
pickle.dump(dict , f)
f.close()

f =open("bfile.dat","rb")
data=pickle.load(f)
StudentRollNo=int(input("Enter the Roll No You want to Search"))

n=len(data['Name'])
i=0
while i<n:    # Iterating through the File to Search the Record
    if data['rollNo'][i] == StudentRollNo:
        print("Record Found")
        print(data['Name'][i],data['rollNo'][i])
        newName = input("Enter the new Name")
        data['Name'][i] = newName  #Update the Record
        print("Record Updated")
        break;
    i=i+1
f.close()

if i==n:  #This will true if not record found
  print("Record Not Found")
else:
    #Again Open file to Write Updated Data
```

```
    f =open("bfile.dat","wb")
    pickle.dump(data,f)
    f.close()
    #Again Open file for Read
    f =open("bfile.dat","rb")
    data=pickle.load(f)
    print("Updated Contents of the Files are")
    print(data)
    f.close()
```

```
Enter the Roll No You want to Search12
Record Found
Laxmi 12
Enter the new NameSanjeev
Record Updated
Updated Contents of the Files are
{'Name': ['Sanjeev', 'Sita', 'Radha'], 'rollNo': [12, 14, 10]}
```

In [34]:
```
#  A PYTHON PROGRAM TO COPY DICTIONARY OBJECT IN BINARY FILE

import pickle
f =open("bfile.dat","wb")
dict = { 'Name' : ['Laxmi','Sita','Radha'],'Age' : [12,14,10] }
pickle.dump(dict , f)
f.close()

f =open("bfile.dat","rb")
data=pickle.load(f)
print(data)

f.close()
```

```
{'Name': ['Laxmi', 'Sita', 'Radha'], 'Age': [12, 14, 10]}
```

In [75]:
```
"""
CSV (stands for comma separated values) format is a commonly used data
format used by spreadsheets. The csv module in Python's standard library
presents classes and methods to perform read/write operations on CSV files.
"""
# Creating a CSV file one row at a time using writerow
import csv
persons=[('Mannat',22,45),('Jai',21,56),('Samidha',20,60)]
csvfile=open('persons.csv','w', newline='')
obj=csv.writer(csvfile)
for person in persons:
    obj.writerow(person)# use of writerow()
csvfile.close()
```

In [76]:
```
# Creating a CSV file all rows at a time using writerows

import csv
persons=[('Arya',22,45),('Jai',21,56),('Samidha',20,60)]
csvfile=open('persons.csv','w', newline='')
obj=csv.writer(csvfile)
obj.writerows(persons) # use of writerows()
csvfile.close()
```

In [77]:
```
# Reading a CSV file rows wise using reader
csvfile=open('persons.csv','r', newline='')
obj=csv.reader(csvfile) # use of reader()
for row in obj:
    print (row)
```

```
['Arya', '22', '45']
['Jai', '21', '56']
['Samidha', '20', '60']
```

In [ ]:
```
"""
Do this assignment yourself
Take  a sample of  ten phishing e-mails (or any  text  file) and
find  most commonly occurring word(s)
"""
```

In [15]:
```
#A PYTHON MYSQL CONNECTIVITY TEST ON MYSQL SERVER
import mysql.connector

#CREATE A CONNECTION
mycon=mysql.connector.connect (host="localhost",user="root",passwd="Admin007%1",  d
atabase="kvplp")
if mycon:
    print("CONNECTION SUCESSFUL")
else:
    print("CONNECTION FAILED")
mycon.close()
```

```
CONNECTION SUCESSFUL
```

In [16]:
```
# MYSQL SHOW DATABASES AND SHOW TABLES COMMAND
import mysql.connector

mycon=mysql.connector.connect (host="localhost",user="root",passwd="Admin007%1")
if mycon:
    cursor=mycon.cursor( )
    cursor.execute("SHOW DATABASES")
    data=cursor.fetchall()
    print(data)

    cursor.execute("USE kvplp")
    print("TABLES UNDER kvplp DATABASE")
    cursor.execute("SHOW TABLES")
    tables=cursor.fetchall()
    print(tables)
    cursor.close()
    mycon.close()
else:
    print("CONNECTION FAILED")
```

```
[('information_schema',), ('TECH',), ('kvplp',), ('mysql',), ('performance_schem
a',), ('sys',)]
TABLES UNDER kvplp DATABASE
[('KVPALAMPUR',), ('X',), ('XI',), ('XII',), ('XIICS',), ('items',)]
```

In [17]:
```
# MYSQL CREATE TABLE COMMAND
import mysql.connector

mycon=mysql.connector.connect (host="localhost",user="root",passwd="Admin007%1",  d
atabase="kvplp")

if mycon:
    cursor=mycon.cursor( )
    cursor.execute("DROP TABLE IF EXISTS XIICS")
    cursor.execute("CREATE TABLE XIICS(ROLLNO INT , NAME VARCHAR(20))")
    cursor.execute("DESC XIICS")
    data=cursor.fetchall()
    print(data)
    cursor.close()
    mycon.close()
else:
    print("CONNECTION FAILED")
```

```
[('ROLLNO', 'int(11)', 'YES', '', None, ''), ('NAME', 'varchar(20)', 'YES', '',
None, '')]
```

In [18]:
```
# MYSQL INSERT COMMAND
import mysql.connector

mycon=mysql.connector.connect (host="localhost",user="root",passwd="Admin007%1",  d
atabase="kvplp")

if mycon:
    cursor=mycon.cursor( )
    cursor.execute("INSERT INTO XIICS VALUES(1,'Bhanu')")
    cursor.execute("INSERT INTO XIICS VALUES(2,'Seema')")
    cursor.execute("INSERT INTO XIICS VALUES(3,'Jaswant')")
    cursor.execute("INSERT INTO XIICS VALUES(4,'Tanu')")
    cursor.execute("COMMIT")

    cursor.execute("SELECT * FROM XIICS")
    data=cursor.fetchall()
    print(data)

    cursor.close()
    mycon.close()

else:
    print("CONNECTION FAILED")
```

```
[(1, 'Bhanu'), (2, 'Seema'), (3, 'Jaswant'), (4, 'Tanu')]
```

In [19]:
```
# MYSQL SELECT COMMAND WITH fetchall()
import mysql.connector

mycon=mysql.connector.connect (host="localhost",user="root",passwd="Admin007%1", da
tabase="kvplp" )
if mycon:
    cursor=mycon.cursor( )
    cursor.execute("SELECT * FROM XIICS")
    data=cursor.fetchall()
    print(data)
    cursor.close()
    mycon.close()
else:
    print("CONNECTION FAILED")
```

```
[(1, 'Bhanu'), (2, 'Seema'), (3, 'Jaswant'), (4, 'Tanu')]
```

In [12]:
```
# MYSQL UPDATE COMMAND
import mysql.connector

#CHANGE THSESE PARAMETERS ACCORDING TO YOUR SETTINGS
mycon=mysql.connector.connect (host="localhost",user="root",passwd="Admin007%1",  d
atabase="kvplp")
if mycon:
    cursor=mycon.cursor( )
    cursor.execute("UPDATE XIICS SET name='Manat' WHERE name='Seema'")

    cursor.execute("SELECT * FROM XIICS")
    data=cursor.fetchall()
    print(data)
    cursor.close()
    mycon.close()
else:
    print("CONNECTION FAILED")
```

```
[(1, 'Bhanu'), (2, 'Manat'), (3, 'Jaswant'), (4, 'Tanu')]
```

In [20]:
```
# MYSQL DELETE COMMAND
import mysql.connector

#CHANGE THSESE PARAMETERS ACCORDING TO YOUR SETTINGS
mycon=mysql.connector.connect (host="localhost",user="root",passwd="Admin007%1",  d
atabase="kvplp")

if mycon:
    cursor=mycon.cursor( )
    cursor.execute("DELETE FROM XIICS WHERE rollno=1")
    cursor.close()

    cursor=mycon.cursor( )
    cursor.execute("SELECT * FROM XIICS")
    data=cursor.fetchall()
    print(data)

    cursor.close()
    mycon.close()
else:
    print("CONNECTION FAILED")
```

```
[(2, 'Seema'), (3, 'Jaswant'), (4, 'Tanu')]
```

In [ ]:
```
"""
Do these SQL assignments yourself.
Create a student table and insert data.
Implement the following SQL commands on the student table:
1.ALTER table to add new attributes / modify data type / drop attribute
2.UPDATE table to modify data
3.ORDER By to display data in ascending / descending order
4.DELETE to remove tuple(s)
5.GROUP BY and find the min, max, sum, count and average
"""
```