1) Python Program to Perform Arithmetic Operations on 2 values

```
a=int(input("Enter the a value"))
b=int(input("Enter the b value"))
c=a+b
print("Sum of 2 numbers is ",c)
d=a-b
print("Difference of 2 numbers is ",d)
e=a*b
print("Product of 2 numbers is ",e)
f=a//b
print("Quotient of ",a, "and",b,"is ",f)
g= a%b
print("Remainder of ",a,"and",b,"is ",g)
```

Output

```
Enter the a value 7
Enter the b value 3
Sum of 2 numbers is  10
Difference of 2 numbers is  4
Product of 2 numbers is  21
Quotient of  7 and 3 is  2
Remainder of  7 and 3 is  1
```

2) Python Program to find Sum and Product of N Numbers

```
n = int(input("Enter the N value"))
sum=0
prod=1
# to find sum of n numbers
for i in range(1,n+1):
   sum = sum + i
# to find product of n numbers
for i in range(1,n+1):
   prod = prod * i
print("Sum of", n, "values is ", sum)
print("Product of", n, "values is ", prod)
```

Output

```
Enter the N value 10
Sum of 10 values is  55
Product of 10 values is  3628800
```

3) Python Program to Check Given Number is Even or Odd

```
n = int(input("Enter N Value"))
if(n%2==0):
   print("Given Number ", n, "is Even")
else:
   print("Given Number ", n, "is Odd")
```

Output

```
Enter N Value 7
Given Number  7 is Odd
Enter N Value 8
Given Number  8 is Even
```

4) Python Program to Print Even and Odd Numbers from 1 to N

```
n = int(input("Enter N Value"))
for i in range(1,n+1):
   if(i%2==0):
      print(i ,"is Even ")
   else:
      print(i, "is Odd ")
```

Output

```
Enter N Value10
1 is a Odd Number
2 is a Even Number
3 is a Odd Number
4 is a Even Number
5 is a Odd Number
6 is a Even Number
7 is a Odd Number
8 is a Even Number
9 is a Odd Number
10 is a Even Number
```

5) Python Program to Print Number of Even and Odd Numbers from 1 to N

```
n = int(input("Enter N Value"))
ecount=ocount=0
for i in range(1,n+1):
    if(i%2==0):
        ecount=ecount+1
    else:
        ocount=ocount+1
print("Number of Even Numbers from 1 to",n," is ",
ecount)
print("Number of Odd Numbers from 1 to ",n," is ",
ocount)
```

Output

Enter N Value 10

Number of Even Numbers from 1 to 10 is 5

Number of Odd Numbers from 1 to 10 is 5

6) Python Program to Print Sum of Even and Odd Numbers from 1 to N

```
1  n = int(input("Enter N Value"))
2  esum=osum=0
3  for i in range(1,n+1):
4      if(i%2==0):
5          esum=esum+i
6      else:
7          osum=osum+i
8  print("Sum of Even Numbers from 1 to",n,"is ",esum)
9  print("Sum of Odd Numbers from 1 to ",n,"is ",osum)
```

Output

Enter N Value 10

Sum of Even Numbers from 1 to 10 is 30

Sum of Odd Numbers from 1 to 10 is 25

7) Python Program to Illustrate Functions

Function With No Argument

```
1  # function definition
2  def greet():
3      print("hello, smith")
4  #calling function
5  greet()
```

OUTPUT

Hello, smith

Function With No Argument

```
1  # function definition
2  def greet(x):
3      print("hello,",x)
4  #calling function
5  name = input("Enter the name")
6  greet(name)
```

OUTPUT

Enter the name Ivan

Hello, Ivan

Function With Default Argument

```
1  # function with default arguments
2  def sum(a,b=3):
3      c=a+b
4      return c
5  # calling the function
6  res1 = sum(2)
7  print("sum of 2 numbers",res1)
8  res2 = sum(4,5)
9  print("sum of 2 numbers",res2)
```

OUTPUT

```
sum of 2 numbers 5
sum of 2 numbers 9
```

Function With Keyword Argument

```
1  # function with Keyword arguments
2  def greet(name,message):
3      print("hello",name,message)
4  greet("Alice","How are you") # positional Arguments
5  greet(message = "how are u",name="jones") # Keyword Argument
```

OUTPUT

```
hello Alice How are you
hello jones how are u
```

Function With Arbitrary Keyword Argument

Code

```
1  # function with variable length arguments
2  def MyFun(*argv):
3      for i in argv:
4          print(i)
5  MyFun('geeks','for','geeks')
```

Code

```
1  # function with variable length Keyword arguments
2  def MyFun(**kwargs):
3      for key,value in kwargs.items():
4          print(key,"=",value)
5  MyFun(first="geeks",mid="for",last="geeks")
```

Output

```
geeks
for
geeks
```

Output

```
first = geeks
mid = for
last = geeks
```

8) Python Program to Print Sum of Individual Digits of the Given Number

```
def sumOfIndividualDigits(n):
    s=0
    while n>0:
        r=n%10
        s=s+r
        n=n//10
    return s;
n = int(input("Enter the N Value"))
res = sumOfIndividualDigits(n)
print("Sum of Individual Digits of the gven Number is ", res)
```

Output

Enter the N Value 176

Sum of Individual Digits of the gven Number is 14

9) Python Program Check the Given Number is Prime or not

```
1  # function to check given number is prime or not
2  def PrimeOrNot(n):
3      c=0
4      for i in range(1,n+1):
5          if(n%i==0):
6              c=c+1
7      if(c==2):
8          print("Given Number is a prime Number")
9      else:
10         print("Given number is not a Prime Number")
11 # main program
12 n = int(input("enter N value"))
13 PrimeOrNot(n)
```

Output

enter N value 5

Given Number is a prime Number

enter N value 6

Given number is not a Prime Number

10) Python Program to print prime numbers from 1 to N

```
# function to check given number is prime or not
def Primes(n):
    for i in range(1,n+1):
        c=0
        for j in range(1,i+1):
            if(i%j==0):
                c=c+1
        if(c==2):
          print(i," is a Prime Number")
# main program
n = int(input("enter N value"))
Primes(n)
```

Output

2 is a Prime Number

3 is a Prime Number

5 is a Prime Number

7 is a Prime Number

11) Python Program to Check the given number is Armstrong or Not

```
1  def Armstrong(n):
2      s=0
3      while n>0:
4          r=n%10
5          s=s+(r*r*r)
6          n=n//10
7      return s;
8  n = int(input("Enter the N Value"))
9  original = n
10 res = Armstrong(n)
11 if (original == res):
12     print(original, "is a Armstrong Number")
13 else:
14     print(original, "is not a Armstrong Number")
```

Output

Enter the N Value 153

153 is a Armstrong Number

Enter the N Value 123

123 is not a Armstron

12) Python Program to print the given number is Reverse

```python
def reverse_number(number):
    reversed_num = 0
    while number != 0:
        last_digit = number % 10
        reversed_num = reversed_num * 10 + last_digit
        number //= 10
    return reversed_num

# Example usage:
number = int(input("Enter a number: "))
result = reverse_number(number)
print("Reversed number:", result)
```

Output

Enter a number: 234

Reversed number: 432

13) Python Program to check the given number is perfect or not

```
1  def is_perfect_number(number):
2      divisors_sum = 0
3      for i in range(1, number):
4          if number % i == 0:
5              divisors_sum += i
6      return divisors_sum
7  # Example usage:
8  number = int(input("Enter a number: "))
9  div_sum = is_perfect_number(number)
10 if (div_sum == number):
11     print(number, "is a perfect number.")
12 else:
13     print(number, "is not a perfect number.")
```

Output

Enter a number: 8

8 is not a perfect number.

Enter a number: 6

6 is a perfect number.

14) Python Program to print nth Fibonacci Number

```
1   f1=1
2   f2=1
3   print("f1 =",f1)
4   print("f2 =",f2)
5   n = int(input("Enter N Value"))
6   for i in range(3,n+1):
7       f3 = f1+f2
8       print(f3)
9       f1=f2
10      f2=f3
11  print(n,"th fibonacci number is",f3)
```

Output

f1 = 1

f2 = 1

Enter N Value 8

2

3

5

8

13

21

8 th fibonacci number is 21

15) Program to check the given number is Harshad Number or not

```
1   def harshad_no(n):
2       temp = n
3       s=0
4       while n>0:
5           m=n%10
6           s=s+m
7           n=n//10
8       if (temp%s==0):
9           print("The Given Number is a Hashad Number")
10      else:
11          print("The Given Number is not a Hashad Number")
12
13  #main program
14  n= int(input("Enter the n value"))
15  harshad_no(n)
```

Output

Enter the n value 81

The Given Number is a Hashad Number

Enter the n value52

The Given Number is not a Hashad Number

16) Program to Illustrate Python List and its void methods

```
# creating a List
l1 =[2,3,4]
print("original List",l1)

# Appending an Element at the end of the list
l1.append(6)
print("After Appending List",l1)

# Appending a list of Elements at the end of the list
l1.extend([8,9,10])
print('After extending the list',l1)

# to insert an element at 3rd position
l1.insert(3,20)
print('after inserting element at 3 rd position',l1)

# sort method
print("Before sorting",l1)
l1.sort()
print("After sorting",l1)

# removing an element at 1st position
del l1[1]
print('after removing element at 1st pos',l1)

# removing an element at 5th position
removed = l1.pop(5)
print("removed element",removed)
print("after removing element 5th element ",l1)

# modifying an element at 3rd position
l1[3] = "Korth"
print("After modifying ",l1)

# modifying elements from 1st to 3rd position
l1[1:3] = "hello","hai","good"
print("after modifying",l1)
```

Output

original List [2, 3, 4]

After Appending List [2, 3, 4, 6]

After extending the list [2, 3, 4, 6, 8, 9, 10]

after inserting element at 3 rd position [2, 3, 4, 20, 6, 8, 9, 10]

Before sorting [2, 3, 4, 20, 6, 8, 9, 10]

After sorting [2, 3, 4, 6, 8, 9, 10, 20]

after removing element at 1st pos [2, 4, 6, 8, 9, 10, 20]

removed element 10

after removing element 5th element [2, 4, 6, 8, 9, 20]

After modifying [2, 4, 6, 'Korth', 9, 20]

after modifying [2, 'hello', 'hai', 'good', 'Korth', 9, 20]

17) Program to Illustrate Python List and its return type methods

```
1   # creating a list
2   l2 = [20,34,67,23,67,78]
3   print("Original List ",l2)
4
5   # remove() method
6   rem = l2.pop()
7   print("removed element",rem)
8   print("after Removing an element from the list are",l2)
9
10  # count() method
11  c = l2.count(67)
12  print("Number of elements in l2 are",c)
13
14  #index() method
15  pos = l2.index(23)
16  print("Position of 23 in the list is ",pos)
```

Output

removed element 78

after Removing an element from the list are [20, 34, 67, 23, 67]

Number of elements in l2 are 2

Position of 23 in the list is 3

18) Program to Illustrate Python List and its Functions

```
l1=[34,56,34,12,78,10]
print("Original List elements are",l1)
print('No.of Elements in the list is',len(l1))
sorted = sorted(l1)
print('original List',l1,'sorted list',sorted)
print("Smallest Element in the list is ",min(l1))
print("Largest Element in the list is ",max(l1))
print("Sum of elements in the list is ",sum(l1))
print("Reverse Order of elements in the list is ", list(reversed(l1)))
```

Output

Original List elements are [34, 56, 34, 12, 78, 10]

No.of Elements in the list is 6

original List [34, 56, 34, 12, 78, 10] sorted list [10, 12, 34, 34, 56, 78]

Smallest Element in the list is 10

Largest Element in the list is 78

Sum of elements in the list is 224

Reverse Order of elements in the list is [10, 78, 12, 34, 56, 34]

19) Program to Illustrate Python tuples

```
t1 =('hello','hai','smith','jones',3,4,5,6,7)
print('Original List',t1)
# Adding elements to tuple. As it is immutable,need to convert into list and then append
y =list(t1)
y.append("orange")
t1 =tuple(y)
print('''After Adding the element 'orange' to the original tuple''',t1)
#Adding a tuple to another tuple
t2 = ("apple","banana","cherry")
print(" new tuple",t2)
t2 += t1
print("after concatenating 2 tuples",t2)
# Modifying the tuple
print("Before modifying the tuple elements",t1)
y =list(t1)
y[1] ="kiwi"
t1 =tuple(y)
print("After modifying the 1st position element",t1)
# Accessing the elements of original tuple
print("elements of the original tuple are ")
for i in t1:
   print(i)
# accessing 3rd element
print('3rd element of the tuple is ',t1[3])
#Accessing  elements from 2nd to 5th
print('from 2nd to 6th ',t1[2:6])
# Accessing last 3 elements
print('Accessing last 3 elements',t1[-1:-4:-1])
#Accessing elements from first to last
print('Accessing from first to last',t1[1:])
```

Output

Original List

('hello', 'hai', 'smith', 'jones', 3, 4, 5, 6, 7)

After Adding the element 'orange' to the original tuple

('hello', 'hai', 'smith', 'jones', 3, 4, 5, 6, 7, 'orange')

new tuple

('apple', 'banana', 'cherry')

after concatenating 2 tuples

('apple', 'banana', 'cherry', 'hello', 'hai', 'smith', 'jones', 3, 4, 5, 6, 7, 'orange')

Before modifying the tuple elements

('hello', 'hai', 'smith', 'jones', 3, 4, 5, 6, 7, 'orange')

After modifying the 1st position element

('hello', 'kiwi', 'smith', 'jones', 3, 4, 5, 6, 7, 'orange')

elements of the original tuple are

hello
kiwi
smith
jones
3
4
5
6
7
Orange

3rd element of the tuple is jones

from 2nd to 6th

('smith', 'jones', 3, 4)

Accessing last 3 elements

('orange', 7, 6)

Accessing from first to last

('kiwi', 'smith', 'jones', 3, 4, 5, 6, 7, 'orange')

20) Program to Illustrate Python tuple methods and Functions

```
t1= (1,2,3,4,9,2,10,2)
print(" Original Tuple",t1)
print("Number of Occurences of element 2 in the original tuple is ",t1.count(2))
print("Position of the element 30 in the given tuple is ", t1.index(9))
print("Number of Elements in the tuple is ",len(t1))
print("Biggest Number in the Tuple is",max(t1))
print("Smallest Number in the Tuple is",min(t1))
print("Sum of the elements of the tuple is",sum(t1))
print("Sorting Tuple Elements ",sorted(t1))
```

Output

Original Tuple (1, 2, 3, 4, 9, 2, 10, 2)

Number of Occurrences of element 2 in the original tuple is 3

Position of the element 30 in the given tuple is 4

Number of Elements in the tuple is 8

Biggest Number in the Tuple is 10

Smallest Number in the Tuple is 1

Sum of the elements of the tuple is 33

21) Program to Illustrate Python Dictionary

```
1   d = {1:"Monday",2:"Tuesday",3:"Wednsday"}
2   print("Original Dictionary",d)
3   d[4] = "Thursday"
4   print("after appending new key value ",d)
5   d[6] = "saturday"
6   print("after appending new element",d)
7   k = d.keys()
8   print("keys of the original dictionary are",k)
9   v = d.values()
10  print("values of the dictionary are",v)
11  v = d.pop(2)
12  print("popped value of the key 2 ",v)
13  print("after popping",d)
```

Output

Original Dictionary {1: 'Monday', 2: 'Tuesday', 3: 'Wednsday'}

after appending new key value {1: 'Monday', 2: 'Tuesday', 3: 'Wednsday', 4: 'Thursday'}

after appending new element {1: 'Monday', 2: 'Tuesday', 3: 'Wednsday', 4: 'Thursday', 6: 'saturday'}

keys of the original dictionary are dict_keys([1, 2, 3, 4, 6])

values of the dictionary are dict_values(['Monday', 'Tuesday', 'Wednsday', 'Thursday', 'saturday'])

popped value of the key 2 Tuesday

after popping {1: 'Monday', 3: 'Wednsday', 4: 'Thursday', 6: 'saturday'}

22) Program to Illustrate Python Set

```
set1 = {2,"hello",56,78,"hai"}
print("original set elements are",set1)
# to add elements to set
set1.add("new")
print("After Adding new element to the set ",set1)
#removing element
set1.remove("hello")
print('''After removing 'hello' from the set''',set1)
set1.pop()
print("after popping the first element",set1)
s1= {2,3,1,5}
s2= {5,1,9,2}
print("original set 1",s1)
print("Original set 2",s2)
print("s1 U s2 =  ",s1.union(s2))
print("s1-s2 = ",s1.difference(s2))
print("s2-s1 =",s2.difference(s1))
print("s1 intersection s2 = ",s1.intersection(s2))
# packing and unpacking  a tuple
print("packing and unpacking  a tuple")
fruits = ("apple", "banana", "cherry")
(a,b,c) = fruits
print(a)
print(b)
print(c)
print("packing and unpacking  a tuple")
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(a, b, *c) = fruits
print(a)
print(b)
print(c)
```

Output

```
original set elements are {2, 56, 'hello', 78, 'hai'}
After Adding new element to the set {'new', 2, 56, 'hello', 78, 'hai'}
After removing 'hello' from the set {'new', 2, 56, 78, 'hai'}
after popping the first element {2, 56, 78, 'hai'}
original set 1 {1, 2, 3, 5}
Original set 2 {1, 2, 5, 9}
s1 U s2 =  {1, 2, 3, 5, 9}
s1-s2 =  {3}
s2-s1 = {9}
s1 intersection s2 =  {1, 2, 5}
packing and unpacking  a tuple
apple
banana
cherry
packing and unpacking  a tuple
apple
banana
['cherry', 'strawberry', 'raspberry']
```

23) Illustration of Python String manipulations

23.1) Python Program to get a string made of the first 2 and last 2 characters of a given string.

```
1  def string_both_ends(str):
2      return str[0:2] + str[-2:]
3  print(string_both_ends('PythonWorld'))
```

Output

Pyld

23.2) Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.

```
1  def change_char(str1):
2      char = str1[0]
3      str1 = str1.replace(char, '$')
4      str1 = char + str1[1:]
5      return str1
6  print(change_char('restart'))
```

Output

resta$t

23.3) Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string.

```
1  def chars_mix_up(a, b):
2      new_a = b[:2] + a[2:]
3      new_b = a[:2] + b[2:]
4      return new_a + ' ' + new_b
5  print(chars_mix_up('abc', 'xyz'))
```

Output

xyc abz

23.4) Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing', add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

```
def add_string(str1):
  length = len(str1)
  if length > 2:
    if str1[-3:] == 'ing':
      str1 += 'ly'
    else:
      str1 += 'ing'
    return str1
print(add_string('ab'))
print(add_string('abc'))
print(add_string('string'))
```

Output

None

abcing

stringly

23.5) Python program to remove the nth index character from a nonempty string.

```
def remove_char(str, n):
      first_part = str[:n]
      last_part = str[n+1:]
      return first_part + last_part
print(remove_char('Python', 0))
print(remove_char('Python', 3))
print(remove_char('Python', 5))
```

Output

ython

Pyton

Pytho

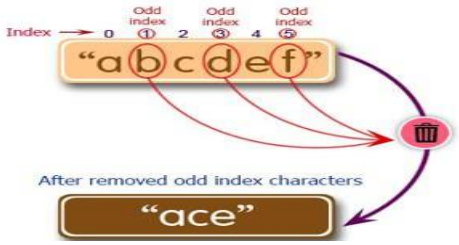
23.6) Write a Python program to change a given string to a newly string where the first and last chars have been exchanged.


```
def change_sring(str1):
    return str1[-1:] + str1[1:-1] + str1[:1]
print(change_sring('abcd'))
```

Output

dbca

23.7) Write a Python program to remove characters that have odd index values in a given string.

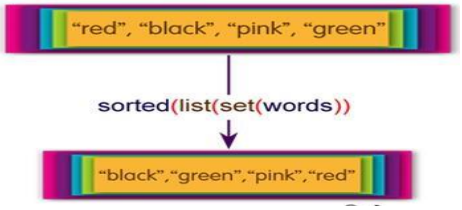



```
def odd_values_string(str):
  result = ""
  for i in range(len(str)):
    if i % 2 == 0:
      result = result + str[i]
  return result
print("characters at  odd index places are : ",odd_values_string('abcdef'))
```

Output

characters at odd index places are : ace

23.8) Write a Python program that accepts a comma-separated sequence of words as input and prints the distinct words in sorted form (alphanumerically).



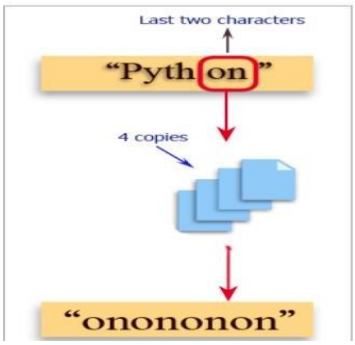

```
items = input("Input comma separated sequence of words")
words_lst = items.split(',')
# convert the list into set to remove duplicate words
words_set = set(words_lst)
# sorting the set values
print(",".join(sorted(words_set)))
```

Output

Input comma eparated sequence of words red,black,pink,green,black

black, green, pink, red

23.9) Write a Python function to get a string made of 4 copies of the last two characters of a speci□ied string (length must be at least 2).

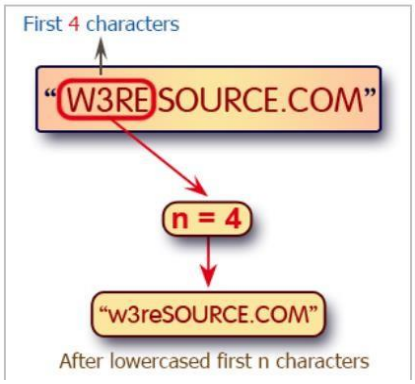



```
def insert_end(str):
    sub_str = str[-2:]
    return sub_str * 4
print(insert_end('Python'))
```

Output

Onononon

23.10) *Python program to lowercase the first n characters in a string.*




```
1  str1 = 'PYTHONWORLD'
2  print(str1[:4].lower() + str1[4:])
```

Output

```
PythONWORLD
```

23.11) *Program to Illustrate the Python String Methods.*

```
str = "welcome to python world"
print("Given String is : ", str)
print("Given String in Upper Case is : ",str.upper())
print("Given String in Lower Case is : ",str.lower())
print("Captitalize first letter of the given string is : ",str.capitalize())
print("After swapping  the case of the given string is : ",str.swapcase())
print('''to check the given string ends with 'world' is :''',str.endswith('world'))
print('''to check the given string ends with 'world' is :''',str.endswith('python'))
print('''to check the given string starts with 'world' is :''',str.startswith('world'))
print('''to check the given string starts with 'world' is :''',str.startswith('welcome'))
print('''No.of times 'e' occurs in the given string is ''',str.count('e'))
print('to check the given string contains all alphabets',str.islower())
```

Output

```
Given String is :  welcome to python world

Given String in Upper Case is :  WELCOME TO PYTHON WORLD

Given String in Lower Case is :  welcome to python world
```

Captitalize first letter of the given string is : Welcome to python world

After swapping the case of the given string is : WELCOME TO PYTHON WORLD

to check the given string ends with 'world' is : True

to check the given string ends with 'world' is : False

to check the given string starts with 'world' is : False

to check the given string starts with 'world' is : True

No.of times 'e' occurs in the given string is 2

to check the given string contains all alphabets True

23.12) Python Program to check whether the given string exists in the Original String

```
1  orig_str="Welcome to Python world"
2  given_str = "to"
3  if given_str in orig_str:
4      print(given_str," is found in the ", orig_str)
5  else:
6      print(given_str," is not found in the" , orig_str)
```

Output

to is found in the Welcome to Python world

23.13) Python Program to replace a word in the given string

```
1  str = "welcome to java world"
2  print(str.replace("java","python"))
```

Output

Welcome to python world

24) Python Program to Illustrate Class-Object

```
class Rectangle:
    def __init__(self,l,b):
        self._l=l
        self.__b=b
    def area(self):
        return self._l*self.__b
r1=Rectangle(2,3)
print('Area of first Rectangle',r1.area())
r2 = Rectangle(4,5)
print('Area of second rectangle',r2.area())
```

Output

Area of first Rectangle 6

Area of second rectangle 20

25) Python Program to Illustrate Constructor Overloading

```
class Const:
    def __init__(self,a=0,b=0,c=0):
        self.a=a
        self.b=b
        self.c=c
    def sum(self):
        return self.a+self.b+self.c
s1 = Const(3,4)
sum2 = s1.sum()
print('sum of 2 numbers',sum2)
s2 = Const(2,3,4)
sum3 = s2.sum()
print('sum of 3 numbers',sum3)
```

Output

sum of 2 numbers is 7

sum of 3 numbers is 9

26) A. Python Program to Illustrate Method Overloading

```
class Meth:
    def sum(self,x=0,y=0,z=0):
        return x+y+z
s1=Meth()
so2=s1.sum(2,3)
so3=s1.sum(1,2,3)
print("sum of 2 numbers is ",so2)
print("sum of 3 numbers is ",so3)
```

Output

```
sum of 2 numbers is  5
sum of 3 numbers is  6
```

27) B. Python Program to Illustrate Method Overloading

```
class methOverloading:
    def add(self, a, b):
        return a + b
    def add(self, a, b, c):
        return a + b + c
    def add(self, *args):
        return sum(args)
# Creating an object of MyClass
obj = methOverloading()
# Calling the overloaded add method
print(obj.add(2, 3))                # Output: 5
print(obj.add(2, 3, 4))             # Output: 9
print(obj.add(2, 3, 4, 5, 6))       # Output: 20
```

Output

```
Sum of 2 Numbers is 5
Sum of 3 Numbers is 9
Sum of 5 Numbers is 20
```

28) Python Program to Illustrate Single Inheritance

```
# Base class
class Animal:
    def __init__(self, name):
        self.name = name

    def make_sound(self):
        pass  # This method will be overridden in the derived
classes

# Derived class inheriting from the Animal class
class Dog(Animal):
    def __init__(self, name, breed):# Call the constructor of
the base class using the super() function
        super().__init__(name)
        self.breed = breed

    def make_sound(self):
        return "Woof!"
# Create instances of the derived classes
dog = Dog("Buddy", "Golden Retriever")

# Accessing attributes and methods of the base class through
the derived classes
print(dog.name ," is a", dog.breed ," and says ",
dog.make_sound())
```

Output

```
Buddy  is a Golden Retriever  and says  Woof!
```

29) Python Program to Illustrate Heriarchical Inheritance

```
class Animal:
    def __init__(self,
        name): self.name
        = name
    def make_sound(self):
        pass
    def move(self):
        print(f"{self.name} is moving.")
class Dog(Animal):
    def make_sound(self):
        print("Woof!")
class Cat(Animal):
    def make_sound(self):
         print("Meow!")
# Creating objects of the derived classes
dog = Dog("Buddy")
cat = Cat("Whiskers")
# Calling the methods
dog.make_sound()   # Output: Woof!
dog.move()         # Output: Buddy is moving.
cat.make_sound()   # Output: Meow!
cat.move()         # Output: Whiskers is moving.
```

Output

Woof!

Buddy is moving

Meow

Whiskers is moving

30) Python Program to Illustrate Multi Level Inheritance

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def introduce(self):
        return  "Hi  my name is ", self.name , " and I am ", self.age ," years old."
# Derived class inheriting from the Person class
class Employee(Person):
    def __init__(self, name, age, employee_id):
        super().__init__(name, age)
        self.employee_id = employee_id
    def work(self):
        return "I am an employee and I am working."
# Further derived class inheriting from the Employee class
class Manager(Employee):
    def __init__(self, name, age, employee_id, department):
        super().__init__(name, age, employee_id)
        self.department = department
    def manage_team(self):
        return "I am managing a team."
# Create an instance of the Manager class
manager = Manager("John Doe", 35, "12345", "Sales")
# Accessing attributes and methods through multilevel inheritance
print(manager.introduce())    # Calls the introduce() method in the Manager class
print(manager.work())         # Calls the work() method in the Employee class
print(manager.manage_team())  # Calls the manage_team() method in the Manager class
```

Output

('Hi my name is ', 'John Doe', ' and I am ', 35, ' years old.')

I am an employee and I am working.

I am managing a team.

31) Python Program to Illustrate Pandas package

```
import pandas as pd
df = pd.read_csv(r"C:\Users\Padmaja
R\OneDrive\Desktop\emp.csv")
print("The complete Employee Data from CSV file")
print(df)

# to get specific  columns
col = input("Enter the column name")
print(df[col])

# to get specific rows
print("Employees who earn salary > 30000")
rows = df.query("SALARY>30000")
print(rows)

# TO GET SUM OF SALARIES OF EMPLOYEES
print("Total salary paid to employees are")
tot_sal = df['SALARY'].sum()
print(tot_sal)

# to get No. of employees in each department
dept_counts = df['Dept.No.'].value_counts()
print(" No.of Employees in Each Department", dept_counts)
```

Output

The Complete Employee Data from CSV ile

ENO.	EMP_NAME	DESIGNATION	SALARY	Dept.No.	AGE
E0001	SMITH	SALES MANAGER	20000	10	32
E0002	JONES	SYSTEM ANALYST	30000	20	26
E0003	KING	MANAGER	50000	10	40
E0004	IVAN	SOFTWARE ENGINEER	37000	20	28
E0005	BAYROSS	SALES MANAGER	26000	10	30
E0006	KORTH	SYSTEM ANALYST	32000	10	35
E0007	LEON	SYSTEM ANALYST	37000	20	29
E0008	MATHEWS	SOFTWARE ENGINEER	30000	10	34
E0009	ALEX	SOFTWARE ENGINEER	28000	10	33
E0010	BOB	SYSTEM ANALYST	35000	20	36

Enter the column name EMP_NAME

EMP_NAME
SMITH
JONES
KING
IVAN
BAYROSS
KORTH
LEON
MATHEWS
ALEX
BOB

Employees who earn salary > 30000

ENO.	EMP_NAME	DESIGNATION	SALARY	Dept.No.	AGE
E0003	KING	MANAGER	50000	10	40
E0004	IVAN	SOFTWARE ENGINEER	37000	20	28
E0006	KORTH	SYSTEM ANALYST	32000	10	35
E0007	LEON	SYSTEM ANALYST	37000	20	29
E0010	BOB	SYSTEM ANALYST	35000	20	36

Total salary paid to employees are

325000

No.of Employees in Each Department Dept.No.

10	**6**
20	**4**