In [17]:
```
"""509. Create a class called NumberSet that accepts 2 integers as input, and defines two
instance variables: num1 and num2, which hold each of the input integers. Then,
create an instance of NumberSet where its num1 is 6 and its num2 is 10. Save this
instance to a variable t
"""
class NumberSet:
    def __init__(self,num1,num2):
        self.num1=num1
        self.num2=num2
    def show(self):
        print(self.num1,self.num2)
```

In [18]:
```
t=NumberSet(6,10)
t.show()
```

```
6 10
```

In [2]:
```
"""510. Create a class called Animal that accepts two numbers as inputs and assigns them
respectively to two instance variables: arms and legs. Create an instance method
called limbs that, when called, returns the total number of limbs the animal has. To
the variable name spider, assign an instance of Animal that has 4 arms and 4 legs. Call
the limbs method on the spider instance and save the result to the variable name
spidlimbs"""
class Animal :
    def __init__(self,arms, legs):
        self.arms = arms
        self.legs = legs
    def limbs (self):
        return self.arms+self.legs
```

In [3]:
```
spider = Animal(4,4)

spidlimbs = spider.limbs()

print(spidlimbs)
```

```
8
```

In [4]:
```
"""511 Write a Python program to create a Vehicle class with max_speed and mileage
instance attributes."""
class Vehicle:
    def __init__(self,name,max_speed,mileage):
        self.name=name
        self.max_speed=max_speed
        self.mileage=mileage
    def vehicle1(self):
        return "hi"
```

In [5]:
```
a=Vehicle("shiinde",24,34)
print(a.name)
print(a.max_speed)
print(a.mileage)
```

```
shiinde
24
34
```

In [6]:
```
""" 512. Write a simple Python class named Student and display its type. Also, display the
__dict__ attribute keys and the value of the __module__ attribute of the Student
class.
"""
class Student:
    pass
print(type(Student))
print(Student.__dict__.keys())
print(Student.__module__)
```

```
<class 'type'>
dict_keys(['__module__', '__dict__', '__weakref__', '__doc__'])
__main__
```

In [7]:
```
"""513. Write a Python class named Student with two attributes student_name, marks.
Modify the attribute values of the said class and print the original and modified values
of the said attributes."""
class Student:
    student_name = 'Terrance Morales'
    marks = 93
print(f"Student Name: {getattr(Student, 'student_name')}")
print(f"Marks: {getattr(Student, 'marks')}")
setattr(Student, 'student_name', 'Angel Brooks')
setattr(Student, 'marks', 95)
print(f"Student Name: {getattr(Student, 'student_name')}")
print(f"Marks: {getattr(Student, 'marks')}")
```

```
Student Name: Terrance Morales
Marks: 93
Student Name: Angel Brooks
Marks: 95
```

In [8]:
```
"""514. Write a Python class named Student with two attributes student_id, student_name.
Add a new attribute student_class. Create a function to display the entire attribute
and their values in Student class."""
class Student:
    student_id = 'V10'
    student_name = 'Jacqueline Barnett'
    def display():
        print(f'Student id: {Student.student_id}\nStudent Name: {Student.student_name}')
print("Original attributes and their values of the Student class:")
Student.display()
```

```
Original attributes and their values of the Student class:
Student id: V10
Student Name: Jacqueline Barnett
```

In [10]:
```
"""515. Write a Python class to implement pow(x, n)"""
class py_solution:
   def pow(self, x, n):
        if x==0 or x==1 or n==1:
            return x

        if x==-1:
            if n%2 ==0:
                return 1
            else:
                return -1
        if n==0:
            return 1
        if n<0:
            return 1/self.pow(x,-n)
        val = self.pow(x,n//2)
        if n%2 ==0:
            return val*val
        return val*val*x
```

In [11]:
```
print(py_solution().pow(2, -3));
print(py_solution().pow(3, 5));
print(py_solution().pow(100, 0));
```

```
0.125
243
1
```

In [12]:
```
"""516. Write a Python class to reverse a string word by word."""
class py_solution:
    def reverse_words(self, s):
        return ' '.join(reversed(s.split()))
```

In [13]:
```
print(py_solution().reverse_words('hello .py'))
```

```
.py hello
```

In [14]:
```
"""517. Write a Python class named Rectangle constructed by a length and width and a
method which will compute the area of a rectangle.
"""
class Rectangle():
    def __init__(self, l, w):
        self.length = l
        self.width  = w

    def rectangle_area(self):
        return self.length*self.width

newRectangle = Rectangle(12, 10)
print(newRectangle.rectangle_area())
```

```
120
```

In [15]:
```
"""518. Write a Python class named Circle constructed by a radius and two methods which will
compute the area and the perimeter of a circle.
"""
class Circle():
    def __init__(self, r):
        self.radius = r

    def area(self):
        return self.radius**2*3.14

    def perimeter(self):
        return 2*self.radius*3.14
```

In [16]:
```
NewCircle = Circle(8)
print(NewCircle.area())
print(NewCircle.perimeter())
```

```
200.96
50.24
```

In [21]:
```
"""Define a class called Bike that accepts a string and a float as input,
and assigns those inputs respectively to two instance variables, color and price.
Assign to the variable testOne an instance of Bike whose color is blue and whose price is 89.99.
Assign to the variable test
Two an instance of Bike whose color is purple and whose price is 25.0."""
class Bike:
    """ Point class for representing and manipulating x,y coordinates. """

    def __init__(self, initX, initY):

        self.color = initX
        self.price = initY
    def show(self):
        print(self.color,self.price)
```

In [22]:
```
testOne = Bike('blue',89.99)
testOne.show()
testTwo = Bike('purple',25.0)
testTwo.show()
```

```
blue 89.99
purple 25.0
```

In [23]:
```
"""Python Program to Create a Class which Performs Basic Calculator Operations"""
class cal():
    def __init__(self,a,b):
        self.a=a
        self.b=b
    def add(self):
        return self.a+self.b
    def mul(self):
        return self.a*self.b
    def div(self):
        return self.a/self.b
    def sub(self):
        return self.a-self.b
a=int(input("Enter first number: "))
b=int(input("Enter second number: "))
obj=cal(a,b)
choice=1
while choice!=0:
    print("0. Exit")
    print("1. Add")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    choice=int(input("Enter choice: "))
    if choice==1:
        print("Result: ",obj.add())
    elif choice==2:
        print("Result: ",obj.sub())
    elif choice==3:
        print("Result: ",obj.mul())
    elif choice==4:
        print("Result: ",round(obj.div(),2))
    elif choice==0:
        print("Exiting!")
    else:
        print("Invalid choice!!")


print()
```

```
Enter first number: 3
Enter second number: 4
0. Exit
1. Add
2. Subtraction
3. Multiplication
4. Division
Enter choice: 1
Result:  7
0. Exit
1. Add
2. Subtraction
3. Multiplication
4. Division
Enter choice: 4
Result:  0.75
0. Exit
1. Add
2. Subtraction
3. Multiplication
4. Division
Enter choice: 2
Result:  -1
0. Exit
1. Add
2. Subtraction
3. Multiplication
4. Division
Enter choice: 3
Result:  12
0. Exit
1. Add
2. Subtraction
3. Multiplication
4. Division
Enter choice: 0
Exiting!
```

In [24]:
```
"""Python Program to Append, Delete and Display Elements of a List using Classes"""
class check():
    def __init__(self):
        self.n=[]
    def add(self,a):
        return self.n.append(a)
    def remove(self,b):
        self.n.remove(b)
    def dis(self):
        return (self.n)

obj=check()

choice=1
while choice!=0:
    print("0. Exit")
    print("1. Add")
    print("2. Delete")
    print("3. Display")
    choice=int(input("Enter choice: "))
    if choice==1:
        n=int(input("Enter number to append: "))
        obj.add(n)
        print("List: ",obj.dis())

    elif choice==2:
        n=int(input("Enter number to remove: "))
        obj.remove(n)
        print("List: ",obj.dis())

    elif choice==3:
        print("List: ",obj.dis())
    elif choice==0:
        print("Exiting!")
    else:
        print("Invalid choice!!")

print()
```

```
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 1
Enter number to append: 23
List:  [23]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 1
Enter number to append: 45
List:  [23, 45]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 1
Enter number to append: 56
List:  [23, 45, 56]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 3
List:  [23, 45, 56]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 2
Enter number to remove: 56
List:  [23, 45]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 3
List:  [23, 45]
0. Exit
1. Add
2. Delete
3. Display
Enter choice: 0
Exiting!
```

In [25]:
```python
"""Python Program to Create a Class in which One Method Accepts a String from the User and Another Prints it"""
class print1():
    def __init__(self):
        self.string=""

    def get(self):
        self.string=input("Enter string: ")

    def put(self):
        print("String is:")
        print(self.string)
```

In [26]:
```python
obj=print1()
obj.get()
obj.put()
```

```
Enter string: Hello Chapter 8
String is:
Hello Chapter 8
```

In [27]:
```
"""Python Program to Create a Class and Get All Possible Distinct Subsets from a Set"""
class sub:
    def f1(self, s1):
        return self.f2([], sorted(s1))

    def f2(self, curr, s1):
        if s1:
            return self.f2(curr, s1[1:]) + self.f2(curr + [s1[0]], s1[1:])
        return [curr]
a=[]
n=int(input("Enter number of elements of list: "))
for i in range(0,n):
    b=int(input("Enter element: "))
    a.append(b)
```

```
Enter number of elements of list: 2
Enter element: 4
Enter element: 5
```

In [28]:
```
print("Subsets: ")
print(sub().f1(a))
```

```
Subsets:
[[], [5], [4], [4, 5]]
```

In [29]:
```
"""Find a pair of elements from a given array whose sum equals a specific target number"""
class py_solution:
  def twoSum(self, nums, target):
       lookup = {}
       for i, num in enumerate(nums):
           if target - num in lookup:
               return (lookup[target - num], i )
           lookup[num] = i
print("index1=%d, index2=%d" % py_solution().twoSum((10,20,10,40,50,60,70),50))
```

```
index1=2, index2=3
```

In [30]:
```
"""Write a Python class to convert a roman numeral to an integer."""
class py_solution:
    def roman_to_int(self, s):
        rom_val = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
        int_val = 0
        for i in range(len(s)):
            if i > 0 and rom_val[s[i]] > rom_val[s[i - 1]]:
                int_val += rom_val[s[i]] - 2 * rom_val[s[i - 1]]
            else:
                int_val += rom_val[s[i]]
        return int_val
```

In [31]:
```
print(py_solution().roman_to_int('MMMCMLXXXVI'))
print(py_solution().roman_to_int('MMMM'))
print(py_solution().roman_to_int('C'))
```

```
3986
4000
100
```

In [32]:
```
"""Write a Python class to convert an integer to a roman numeral"""
class py_solution:
    def int_to_Roman(self, num):
        val = [
            1000, 900, 500, 400,
            100, 90, 50, 40,
            10, 9, 5, 4,
            1
            ]
        syb = [
            "M", "CM", "D", "CD",
            "C", "XC", "L", "XL",
            "X", "IX", "V", "IV",
            "I"
            ]
        roman_num = ''
        i = 0
        while  num > 0:
            for _ in range(num // val[i]):
                roman_num += syb[i]
                num -= val[i]
            i += 1
        return roman_num
```

In [33]:
```
print(py_solution().int_to_Roman(1))
print(py_solution().int_to_Roman(4000))
```

```
I
MMMM
```

In [34]:
```
"""Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. 
These brackets must be close in the correct order, 
for example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid."""
class py_solution:
   def is_valid_parenthese(self, str1):
        stack, pchar = [], {"(": ")", "{": "}", "[": "]"}
        for parenthese in str1:
            if parenthese in pchar:
                stack.append(parenthese)
            elif len(stack) == 0 or pchar[stack.pop()] != parenthese:
                return False
        return len(stack) == 0
```

In [35]:
```
print(py_solution().is_valid_parenthese("(){}[]"))
print(py_solution().is_valid_parenthese("()[{)}"))
print(py_solution().is_valid_parenthese("()"))
```

```
True
False
True
```

In [36]:
```
"""Write a Python class to find the three elements that sum to zero from a set (array) of n real numbers."""
class py_solution:
 def threeSum(self, nums):
        nums, result, i = sorted(nums), [], 0
        while i < len(nums) - 2:
            j, k = i + 1, len(nums) - 1
            while j < k:
                if nums[i] + nums[j] + nums[k] < 0:
                    j += 1
                elif nums[i] + nums[j] + nums[k] > 0:
                    k -= 1
                else:
                    result.append([nums[i], nums[j], nums[k]])
                    j, k = j + 1, k - 1
                    while j < k and nums[j] == nums[j - 1]:
                        j += 1
                    while j < k and nums[k] == nums[k + 1]:
                        k -= 1
            i += 1
            while i < len(nums) - 2 and nums[i] == nums[i - 1]:
                i += 1
        return result
```

In [37]:
```
print(py_solution().threeSum([-25, -10, -7, -3, 2, 4, 8, 10]))
```

```
[[-10, 2, 8], [-7, -3, 10]]
```

In [ ]:
```

```