NAME – RAJDEEP JAISWAL

UID - 20BCS2761

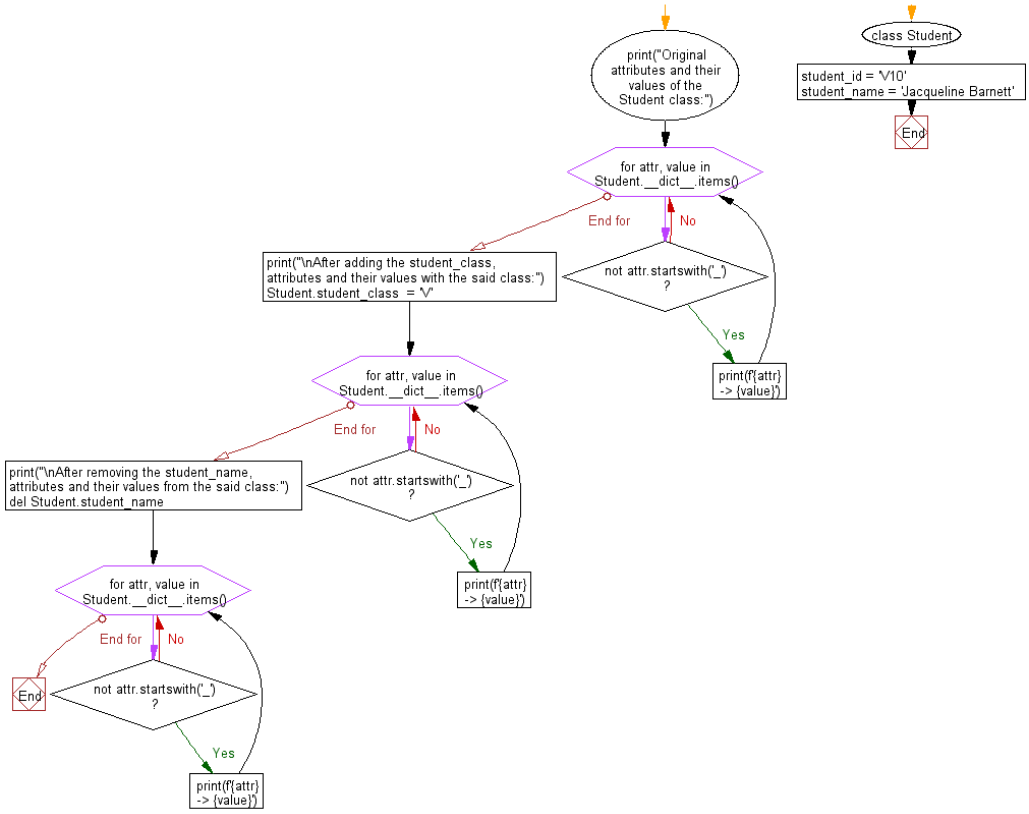
BRANCH – B.TECH CSE

SUBJECT – PYTHON

SEC – 615 - B

1. Aim - Write a Python class named Student with two attributes student_id, student_name. Add a new attribute student_class and display the entire attribute and their values of the said class. Now remove the student_name attribute and display the entire attribute with values.

SOL –
Flowchart/Algo –



CODE IN TEXT –

```
class Student:
    student_id = 'V10'
    student_name = 'Jacqueline Barnett'
print("Original attributes and their values of the Student class:")
```

```
for attr, value in Student.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
print("\nAfter adding the student_class, attributes and their values with the said class:")
Student.student_class  = 'V'
for attr, value in Student.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
print("\nAfter removing the student_name, attributes and their values from the said
class:")
del Student.student_name
#delattr(Student, 'student_name')
for attr, value in Student.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
```

CODE IN COMPILER –

main.py

Run

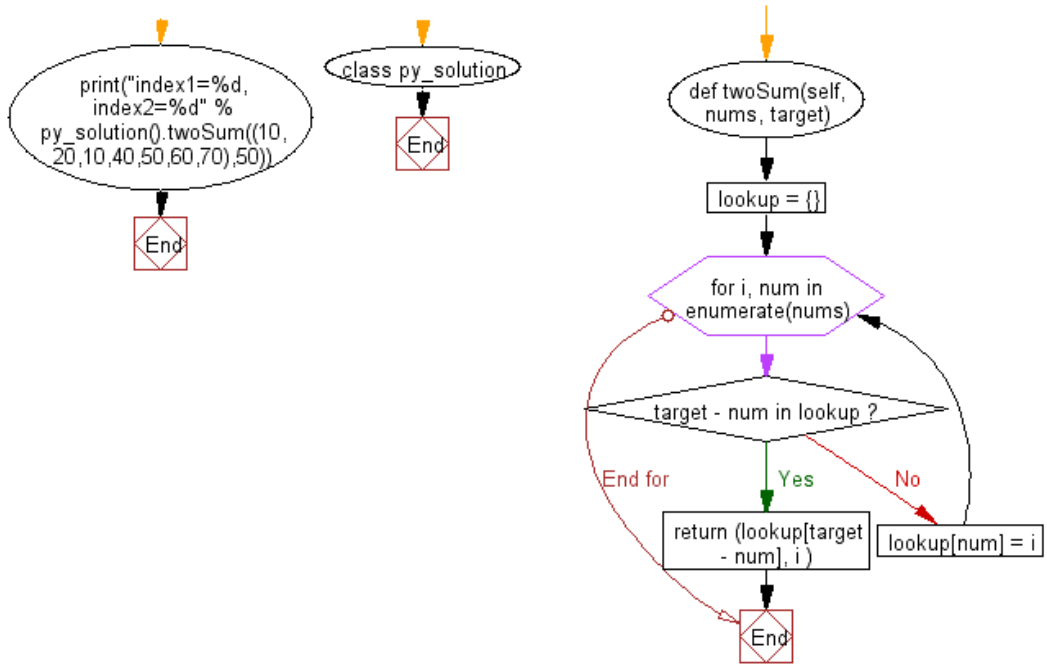
```
class Student:
    student_id = 'V10'
    student_name = 'Jacqueline Barnett'
print("Original attributes and their values of the Student class:")
for attr, value in Student.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
print("\nAfter adding the student_class, attributes and their values
    with the said class:")
Student.student_class  = 'V'
for attr, value in Student.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
print("\nAfter removing the student_name, attributes and their values
    from the said class:")
del Student.student_name
#delattr(Student, 'student_name')
for attr, value in Student.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
```

OUTPUT PROGRAM –

```
Shell                                                    Clear
Original attributes and their values of the Student class:
student_id -> V10
student_name -> Jacqueline Barnett

After adding the student_class, attributes and their values with the said
    class:
student_id -> V10
student_name -> Jacqueline Barnett
student_class -> V

After removing the student_name, attributes and their values from the said
    class:
student_id -> V10
student_class -> V
>
```

2. **Aim** -Write a Python class to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number.

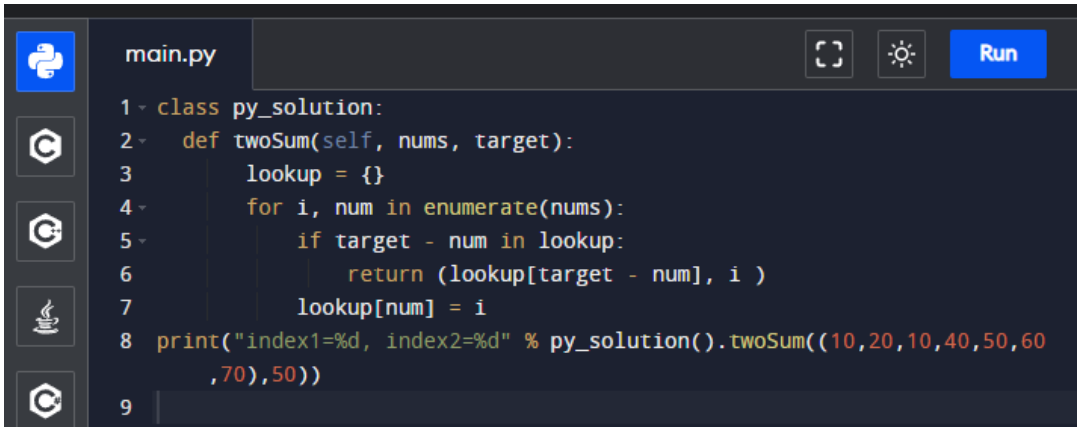
Flowchart – Algo-



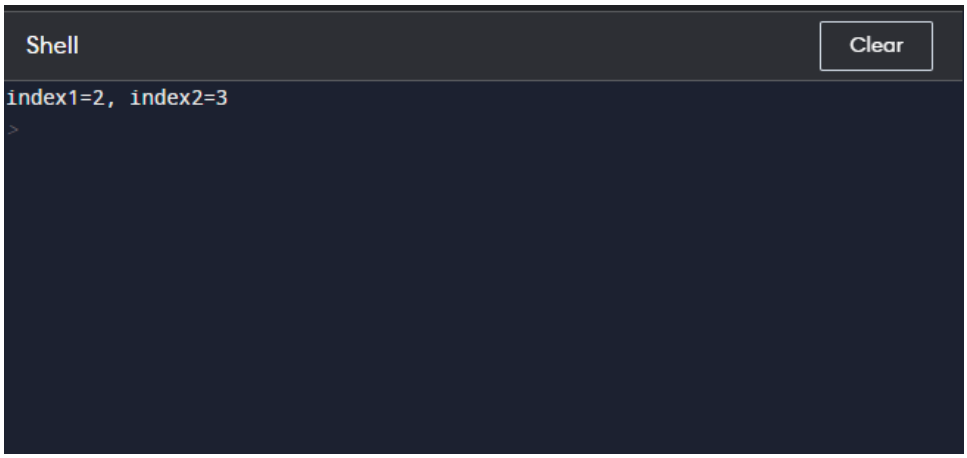
CODE IN TEXT –

```
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))
```

CODE IN COMPILER –

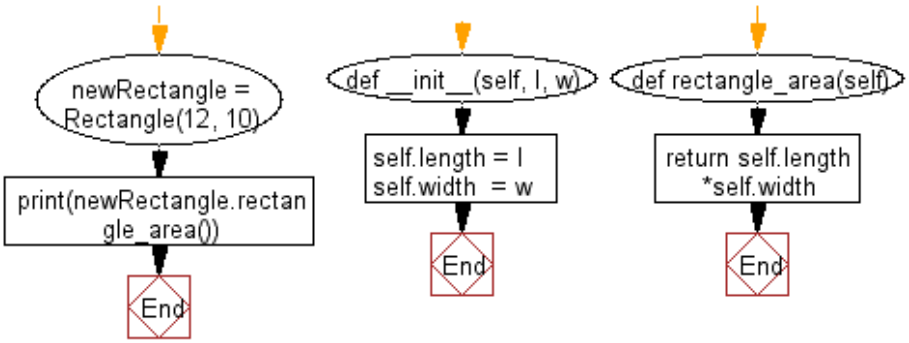


Output program –



3 . Aim - Write a Python class named Rectangle constructed by a length and width and a method which will compute the area of a rectangle

Flowchart – Algo-



CODE IN TEXT –

```
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())
```

CODE IN COMPILER –

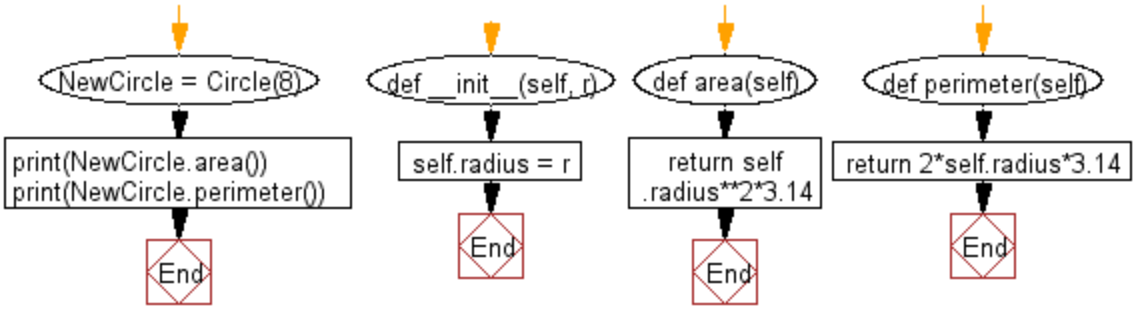
```python
class Rectangle():
    def __init__(self, l, w):
        self.length = l
        self.width  = w

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

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

OUTPUT PROGRAM –

```
120
>
```

4. Aim - Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle

Flowchart – Algo-



CODE IN TEXT –

```
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


NewCircle = Circle(8)

print(NewCircle.area())

print(NewCircle.perimeter())
```

CODE IN COMPILER -

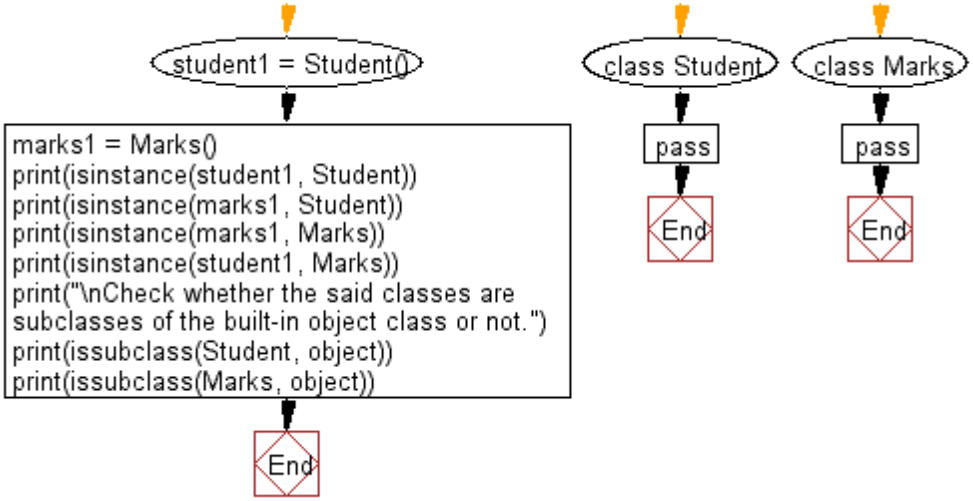
```python
class Circle():
    def __init__(self, r):
        self.radius = r

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

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

NewCircle = Circle(8)
print(NewCircle.area())
print(NewCircle.perimeter())
```

OUTPUT PROGRAM –

```
200.96
50.24
```

5 . Aim - Write a Python program to crate two empty classes, Student and Marks. Now create some instances and check whether they are instances of the said classes or not. Also, check whether the said classes are subclasses of the built-in object class or not

Flowchart – Algo-



CODE IN TEXT –

```
class Student:
    pass
class Marks:
    pass
student1 = Student()
marks1 = Marks()
print(isinstance(student1, Student))
print(isinstance(marks1, Student))
print(isinstance(marks1, Marks))
print(isinstance(student1, Marks))
print("\nCheck whether the said classes are subclasses of the built-in object class or not.")
```

print(issubclass(Student, object))

print(issubclass(Marks, object))

COIDE IN COMPILER –

main.py

```python
class Student:
    pass
class Marks:
    pass
student1 = Student()
marks1 = Marks()
print(isinstance(student1, Student))
print(isinstance(marks1, Student))
print(isinstance(marks1, Marks))
print(isinstance(student1, Marks))
print("\nCheck whether the said classes are subclasses of the built-in object class or not.")
print(issubclass(Student, object))
print(issubclass(Marks, object))
```

OUTPUT PROGRAM –

Shell

```
True
False
True
False

Check whether the said classes are subclasses of the built-in object class
    or not.
True
True
>
```