

NAME - RAJDEEP JAISWAL

UID - 20BCS2761

BRANCH - B.TECH CSE

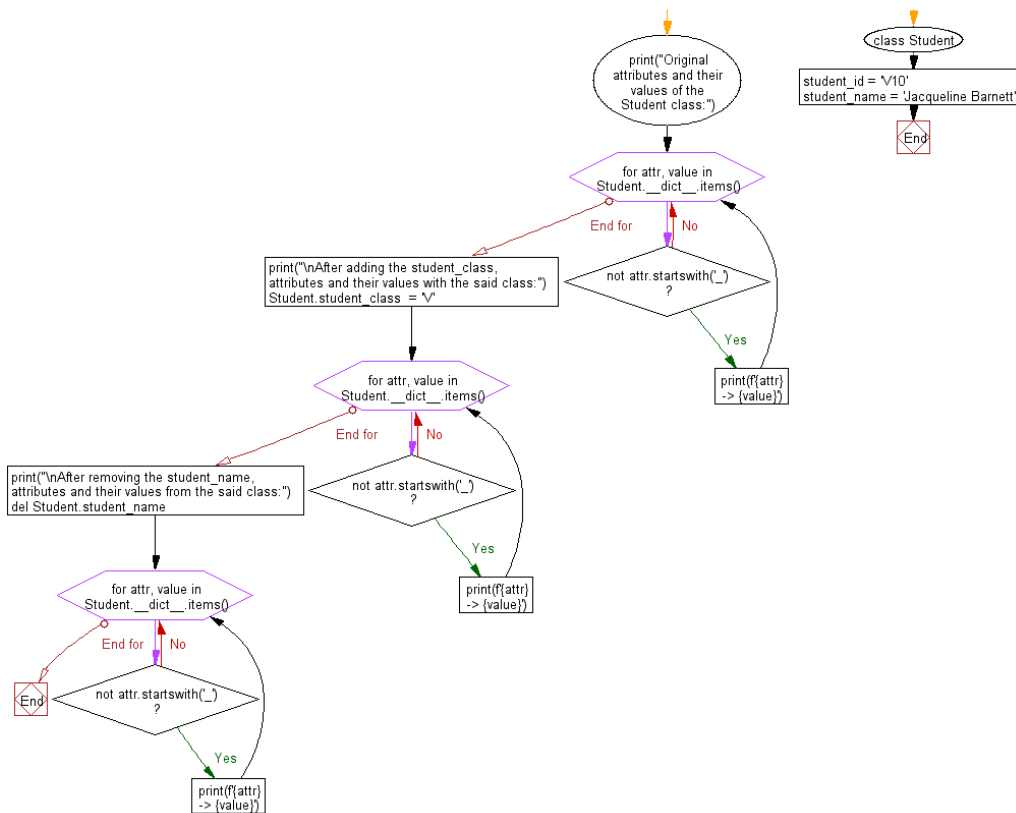
SUBJECT - PYTHON

SEC - 615 - B

- 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 -

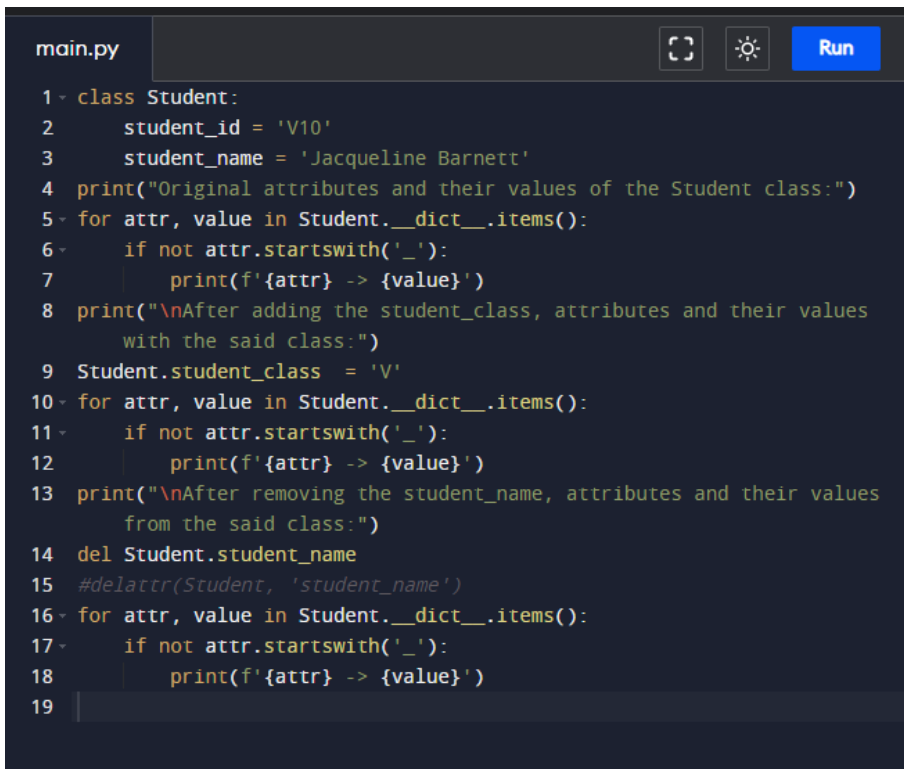
```

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]
1 class Student:
2     student_id = 'V10'
3     student_name = 'Jacqueline Barnett'
4     print("Original attributes and their values of the Student class:")
5     for attr, value in Student.__dict__.items():
6         if not attr.startswith('_'):
7             print(f'{attr} -> {value}')
8     print("\nAfter adding the student_class, attributes and their values
with the said class:")
9     Student.student_class = 'V'
10    for attr, value in Student.__dict__.items():
11        if not attr.startswith('_'):
12            print(f'{attr} -> {value}')
13    print("\nAfter removing the student_name, attributes and their values
from the said class:")
14    del Student.student_name
15    #delattr(Student, 'student_name')
16    for attr, value in Student.__dict__.items():
17        if not attr.startswith('_'):
18            print(f'{attr} -> {value}')
19
```

OUTPUT PROGRAM –

```

Shell
Clear

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

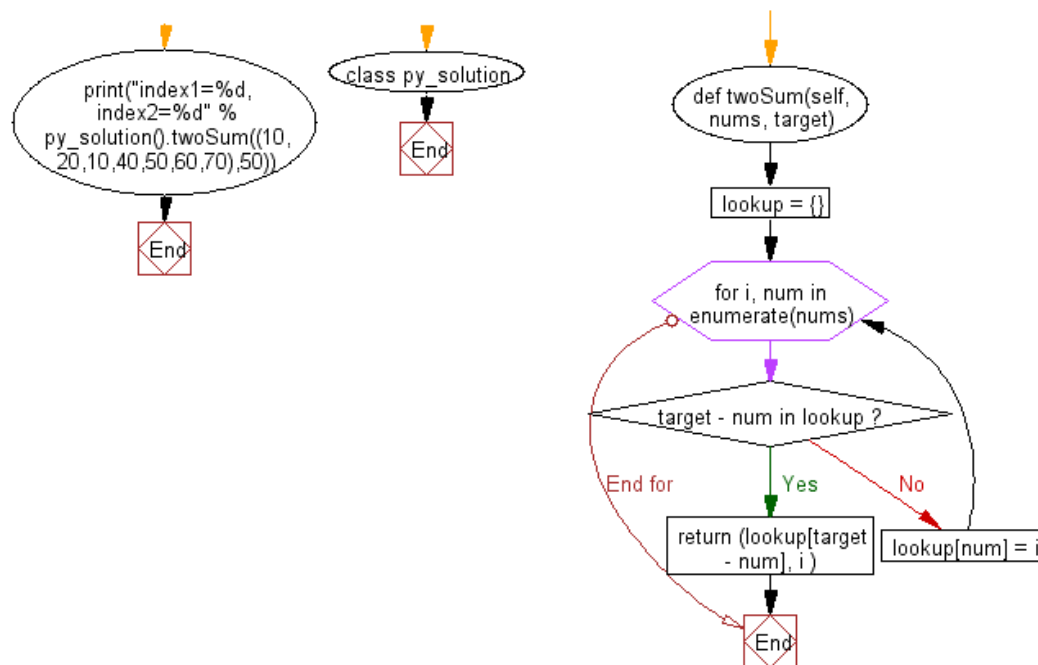
After adding the student_class, attributes and their values with the said class:
student_id -> V10
student_name -> Jacqueline Barnett
student_class -> V

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):
```

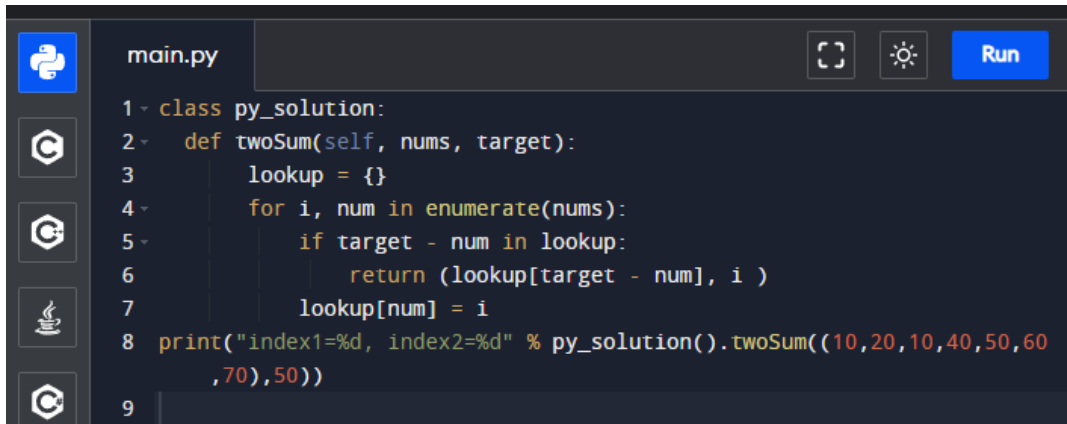
```
        if target - num in lookup:
```

```
            return (lookup[target - num], i )
```

```
        lookup[num] = i
```

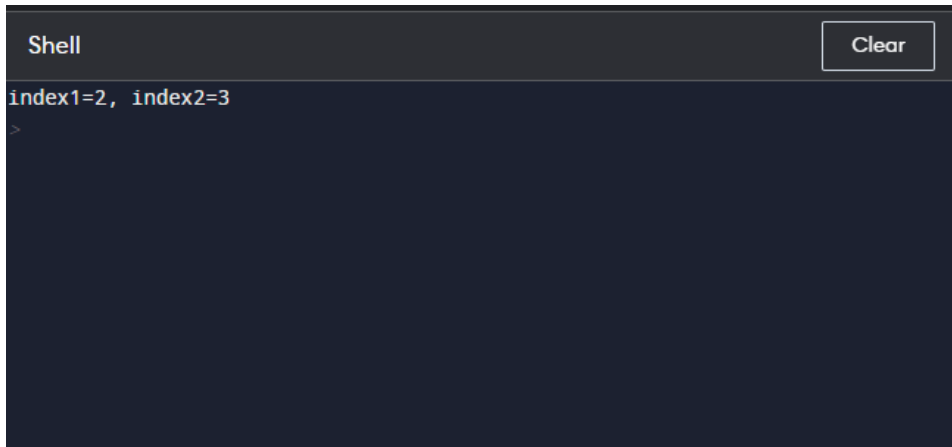
```
print("index1=%d, index2=%d" % py_solution().twoSum((10,20,10,40,50,60,70),50))
```

CODE IN COMPILER –



```
main.py [Run]
1 class py_solution:
2     def twoSum(self, nums, target):
3         lookup = {}
4         for i, num in enumerate(nums):
5             if target - num in lookup:
6                 return (lookup[target - num], i )
7             lookup[num] = i
8     print("index1=%d, index2=%d" % py_solution().twoSum((10,20,10,40,50,60
9         ,70),50))
```

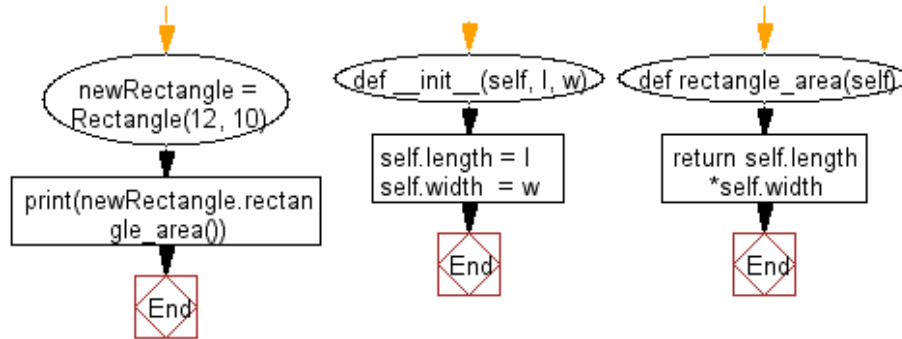
Output program –



```
Shell [Clear]
index1=2, index2=3
>
```

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 –

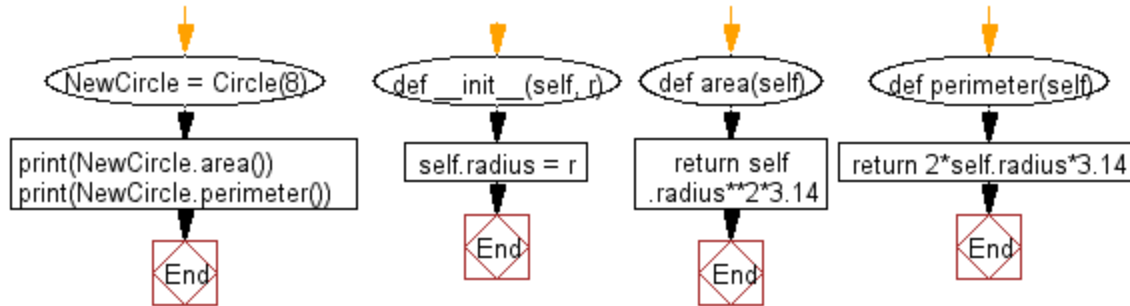
```
main.py ☐ ☐ Run  
1 class Rectangle():  
2     def __init__(self, l, w):  
3         self.length = l  
4         self.width = w  
5  
6     def rectangle_area(self):  
7         return self.length*self.width  
8  
9 newRectangle = Rectangle(12, 10)  
10 print(newRectangle.rectangle_area())  
11
```

OUTPUT PROGRAM –

```
Shell Clear  
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):
```

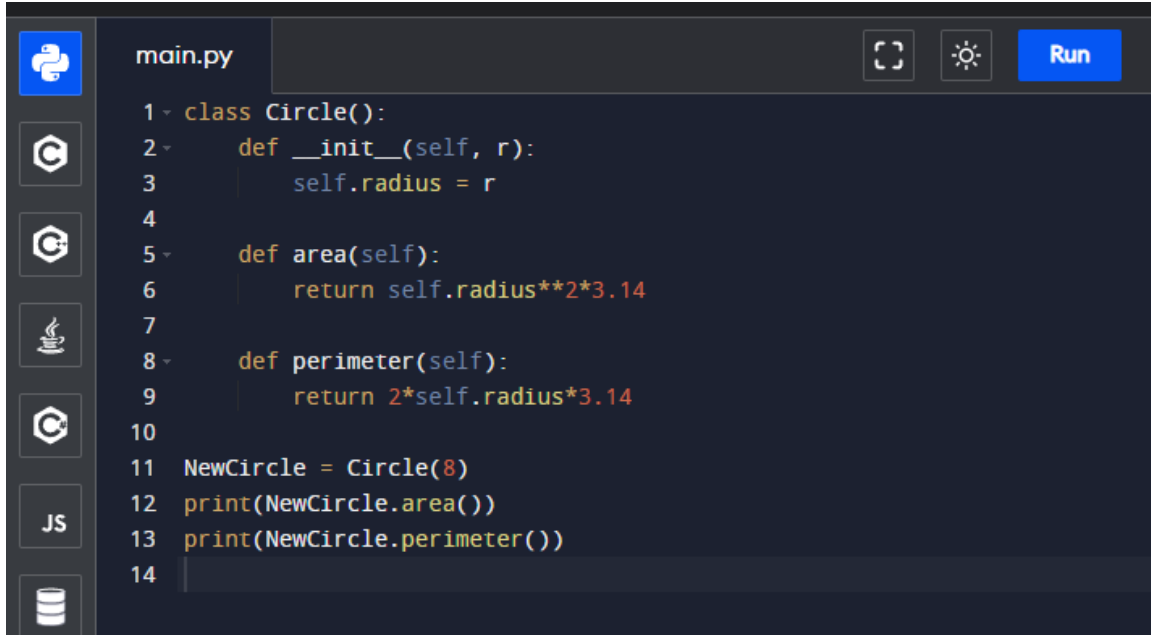
```
        return 2*self.radius*3.14
```

```
NewCircle = Circle(8)
```

```
print(NewCircle.area())
```

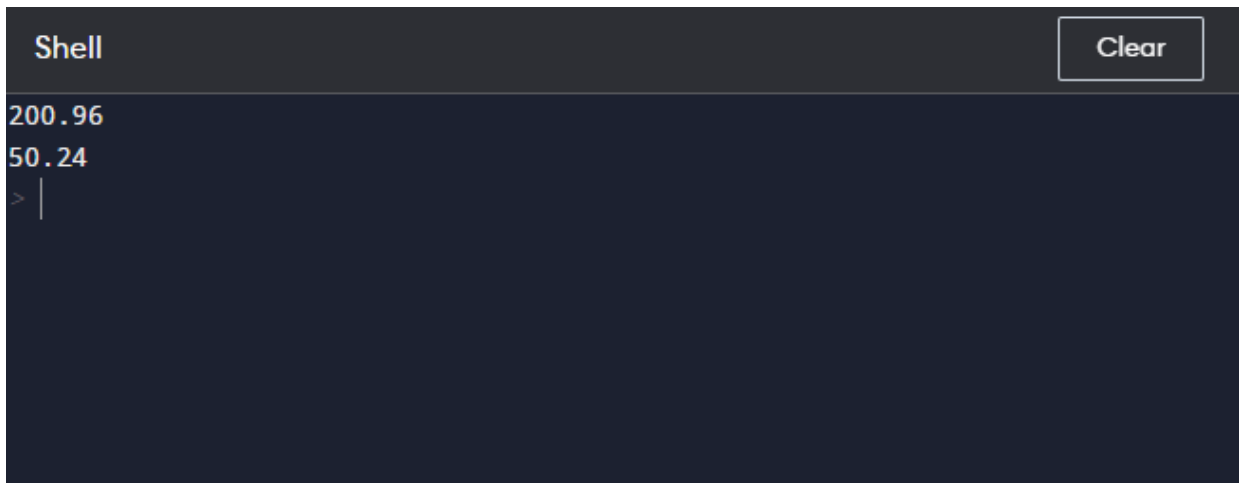
```
print(NewCircle.perimeter())
```

CODE IN COMPILER -



```
main.py ⌵ ⌵ ⌵ Run  
1 class Circle():  
2     def __init__(self, r):  
3         self.radius = r  
4  
5     def area(self):  
6         return self.radius**2*3.14  
7  
8     def perimeter(self):  
9         return 2*self.radius*3.14  
10  
11 NewCircle = Circle(8)  
12 print(NewCircle.area())  
13 print(NewCircle.perimeter())  
14
```

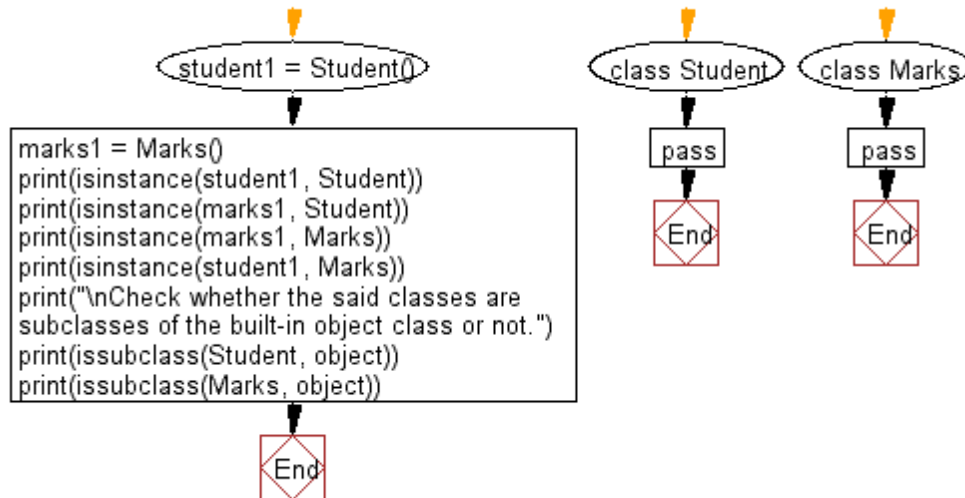
OUTPUT PROGRAM -



```
Shell Clear  
200.96  
50.24  
> |
```


5 . Aim - Write a Python program to create 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))

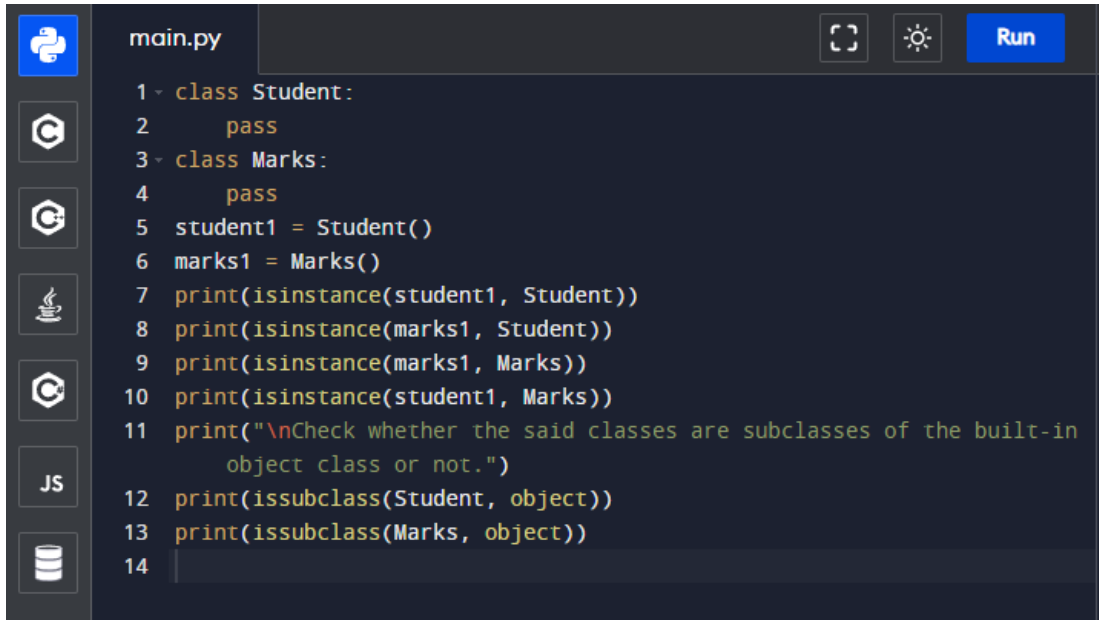
print(isinstance(student1, Marks))

print("\nCheck whether the said classes are subclasses of the built-in object class or not.")

```
print(issubclass(Student, object))
```

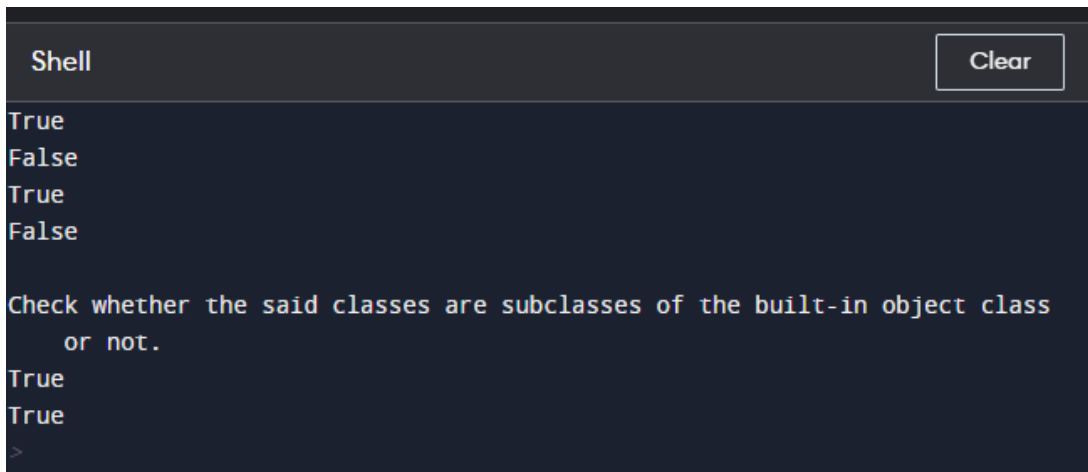
```
print(issubclass(Marks, object))
```

COIDE IN COMPILER –



```
main.py [ ] [ ] [ ] Run  
1 class Student:  
2     pass  
3 class Marks:  
4     pass  
5 student1 = Student()  
6 marks1 = Marks()  
7 print(isinstance(student1, Student))  
8 print(isinstance(marks1, Student))  
9 print(isinstance(marks1, Marks))  
10 print(isinstance(student1, Marks))  
11 print("\nCheck whether the said classes are subclasses of the built-in  
    object class or not.")  
12 print(issubclass(Student, object))  
13 print(issubclass(Marks, object))  
14
```

OUTPUT PROGRAM –



```
Shell [Clear]  
True  
False  
True  
False  
  
Check whether the said classes are subclasses of the built-in object class  
or not.  
True  
True  
>
```