Answers for Debugging Exercises: Chapter 11

Find the Output

1.

```
class Point:
  def __init__(self, x, y):
    self.x = x
    self.y = y
  def __abs__(self):
    return (self.x**2 + self.y**2)**0.5
  def __add__(self, P):
    return Point(self.x + P.x, self.y + P.y)
  def display(self):
    print(self.x, self.y)
P1 = Point(12, 25)
P2 = Point(21, 45)
Print(abs(P2))
P1 = P1+ P2
P1.display()
```

Ans. 49.6588360717

33 70

2.

```
class A(object):
    def __init__(self, num):
        self.num = num
    def __eq__(self, other):
        return self.num == other.num
class B(object):
```

```
        def __init__(self, num):
            self.num = num
print(A(5) == B(5))
```

Ans. True

3.

```
class Circle:
  def __init__(self, radius):
    self.__radius = radius
  def getRadius(self):
    return self.__radius
  def area(self):
    return 3.14 * self.__radius ** 2
  def __add__(self, C):
    return Circle( self.__radius + C.__radius )
C1 = Circle(5)
C2 = Circle(9)
C3 = C1 + C2
print("RADIUS : ",C3.getRadius())
print("AREA : ", C3.area())
```

Ans.

```
RADIUS :  14
AREA :  615.44
```

4.

```
class Circle:
  def __init__(self, radius):
    self.__radius = radius
  def __gt__(self, another_circle):
    return self.__radius > another_circle.__radius
```

```
  def __lt__(self, C):
    return self.__radius < C.__radius
  def __str__(self):
    return "Circle has radius " + str(self.__radius)
C1 = Circle(5)
C2 = Circle(9)
print(C1)
print(C2)
print("C1 < C2 : ", C1 < C2)
print("C2 > C1 : ", C1 > C2)
```

Ans.

```
Circle has radius 5
Circle has radius 9
C1 < C2 :  True
C2 > C1 :  False
```

5.

```
class One:
    def __init__(self):
        num = 10
    def __eq__(self, T):
        if isinstance(T, One):
            return True
        else:
            return NotImplemented
class Two:
    def __init__(self):
        num = 100
print(One() == Two())
```

Ans. False

6.

```
class A:
  def __bool__(self):
    return True
X = A()
if X:
  print('yes')
```

Ans. yes

7.

```
class String(object):
  def __init__(self, val):
    self.val = val
  def __add__(self, other):
    return self.val + '....' + other.val
  def __sub__(self, other):
    return "Not Implemented"
S1 = String("Hello")
S2 = String("World")
print(S1 + S2)
print(S1 - S2)
```

Ans.

```
Hello....World
Not Implemented
```

8.

```
class String(object):
  def __init__(self, val):
```

```
    self.val = val
  def __str__(self):
    return self.val
  def __repr__(self):
    return "This is String representation of " + self.val
S = String("Hi")
print(str(S))
```

Ans.

```
Hi
```

9.

```
class A:
  def __len__(self):
    return 0
X = A()
if not X:
  print('no')

else:
  print('yes')
```

Ans. no

10.

```
class A:
  def __init__(self):
    self.str = "abcdef"
  def __getitem__(self, i):
    return self.str[i]
x = A()
for i in x:
```

```
    print(i,end=" ")
```

Ans. a b c d e f

11.

```
class A:
  str = "Hi"
  def __gt__(self, str):
    return self.str > str
X = A()
print(X > 'hi')
```

Ans. False

Find the Error

1.

```
class Matrix:
    def __init__(self):
        Mat = []
    def setValue(self, number):
        self.number = number
    def display(self):
        print(self.number)
M1 = Matrix()
M1.setValue(([1,2],[3,4]))
M2 = Matrix()
M2.setValue(([5,6],[2,3]))
M3 = Matrix()
M3 = M1 + M2
M3.display()
```

Ans. TypeError: unsupported operand type(s) for +: 'Matrix' and 'Matrix'

2.

```
class A(object):
    def __init__(self, num):
        self.num = num
```

```
        def __eq__(self, other):
            return self.num == other.num
    class B(object):
        def __init__(self, val):
            self.val = val
    print(A(5) == B(5))
```

Ans. AttributeError: 'B' object has no attribute 'num'

3.
```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __mul__(self, num):
        return self.x * num + self.y * num
P1 = Point(3, 4)
print(2*P1)
```

Ans. TypeError: unsupported operand type(s) for *: 'int' and 'Point'

4.
```
class String(object):
    def __init__(self, val):
        self.val = val
S1 = String("Hello")
print(S1[5])
```

Ans. TypeError: 'String' object does not support indexing

5.
```
class Number:
  def __init__(self, num):
    self.num = num
  def __sub__(self, N):
    return Number(self.num - N)
  def __sub__(N, self):
    return Number(N - self.num)
x = Number(4)
y = x-4
```

Ans. AttributeError: 'int' object has no attribute 'num'

6.
```
class A:
    def __init__(self):
      self.str = "abcdef"
    def __setitem__(self, i, val):
      self.str[i] = val
x = A()
x[2] = 'X'
```

Ans. TypeError: 'str' object does not support item assignment