Course Title: Programming Language II
Course Code: CSE111
Lab Assignment no: 8

Task - 1

Let’s Play with **Numbers**!!!

Write the **ComplexNumber** class so that the following code generates the output below.

```python
class RealNumber:

    def __init__(self, r=0):
        self.__realValue = r
    def getRealValue(self):
        return self.__realValue
    def setRealValue(self, r):
        self.__realValue = r
    def __str__(self):
        return 'RealPart: '+str(self.getRealValue())


cn1 = ComplexNumber()
print(cn1)
print('---------')
cn2 = ComplexNumber(5,7)
print(cn2)
```

OUTPUT:

```
RealPart: 1.0
ImaginaryPart: 1.0
--------------------
RealPart: 5.0
ImaginaryPart: 7.0
```

Task - 2 [Ungraded - Optional]

Write the **ComplexNumber** class so that the following code generates the output below.

```
class RealNumber:
    def __init__(self, number=0):
        self.number = number
    def __add__(self, anotherRealNumber):
        return self.number + anotherRealNumber.number
    def __sub__(self, anotherRealNumber):
        return self.number - anotherRealNumber.number
    def __str__(self):
        return str(self.number)

r1 = RealNumber(3)
r2 = RealNumber(5)
print(r1+r2)
cn1 = ComplexNumber(2, 1)
print(cn1)
cn2 = ComplexNumber(r1, 5)
print(cn2)
cn3 = cn1 + cn2
print(cn3)
cn4 = cn1 - cn2
print(cn4)
```

OUTPUT:

```
8
2 + 1i
3 + 5i
5 + 6i
-1 - 4i
```

Task - 3

Write the **CheckingAccount** class so that the following code generates the output below:

```
class Account:
    def __init__(self, balance):
        self._balance = balance

    def getBalance(self):
        return self._balance



print('Number of Checking Accounts: ', CheckingAccount.numberOfAccount)
print(CheckingAccount())
print(CheckingAccount(100.00))
print(CheckingAccount(200.00))
print('Number of Checking Accounts: ', CheckingAccount.numberOfAccount)
```

OUTPUT:

```
Number of Checking
Accounts: 0
Account Balance: 0.0
Account Balance: 100.00
Account Balance: 200.00
Number of Checking
Accounts: 3
```

Task - 4

Write the **Mango** and the **Jackfruit** classes so that the following code generates the output below:

```
class Fruit:
    def __init__(self, formalin=False, name=''):
        self.__formalin = formalin
        self.name = name

    def getName(self):
        return self.name

    def hasFormalin(self):
        return self.__formalin

class testFruit:
    def test(self, f):
        print('----Printing Detail----')
        if f.hasFormalin():
            print('Do not eat the',f.getName(),'.')
            print(f)
        else:
            print('Eat the',f.getName(),'.')
            print(f)

m = Mango()
j = Jackfruit()
t1 = testFruit()
t1.test(m)
t1.test(j)
```

OUTPUT:

```
----Printing Detail-----
Do not eat the Mango.
Mangos are bad for you
----Printing Detail-----
Eat the Jackfruit.
Jackfruits are good for you
```

Task - 5

Write the **ScienceExam** class so that the following code generates the output below:

```python
class Exam:
    def __init__(self,marks):
        self.marks = marks
        self.time = 60

    def examSyllabus(self):
        return "Maths , English"
    def examParts(self):
        return "Part 1 - Maths\nPart 2 - English\n"


engineering = ScienceExam(100,90,"Physics","HigherMaths")
print(engineering)
print('-----------------------------------')
print(engineering.examSyllabus())
print(engineering.examParts())
print('===================================')
architecture =
ScienceExam(100,120,"Physics","HigherMaths","Drawing")
print(architecture)
print('-----------------------------------')
print(architecture.examSyllabus())
print(architecture.examParts())
```

OUTPUT:

```
Marks: 100 Time: 90 minutes Number of
Parts: 4
-----------------------------------
Maths , English , Physics , HigherMaths
Part 1 - Maths
Part 2 - English
Part 3 - Physics
Part 4 - HigherMaths
===================================
Marks: 100 Time: 120 minutes Number of
Parts: 5
-----------------------------------
Maths , English , Physics , HigherMaths
, Drawing
Part 1 - Maths
Part 2 - English
Part 3 - Physics
Part 4 - HigherMaths
Part 5 - Drawing
```

Task - 6

Given the following class, write the code for the **Sphere** and the **Cylinder** class so that the following output is printed.

```
class Shape3D:

  pi = 3.14159
  def __init__(self, name = 'Default', radius = 0):
    self._area = 0
    self._name = name
    self._height = 'No need'
    self._radius = radius

  def calc_surface_area(self):
    return 2 * Shape3D.pi * self._radius

  def __str__(self):
      return "Radius: "+str(self._radius)


sph = Sphere('Sphere', 5)
print('----------------------------------')
sph.calc_surface_area()
print(sph)
print('==================================')
cyl = Cylinder('Cylinder', 5, 10)
print('----------------------------------')
cyl.calc_surface_area()
print(cyl)
```

OUTPUT:

```
Shape name: Sphere, Area Formula: 4 * pi * r
* r
----------------------------------
Radius: 5, Height: No need
Area: 314.159
==================================
Shape name: Cylinder, Area Formula: 2 * pi *
r * (r + h)
----------------------------------
Radius: 5, Height: 10
Area: 471.2385
```

Task - 7

Write the **PokemonExtra** class so that the following code generates the output below:

```
class PokemonBasic:

  def __init__(self, name = 'Default', hp = 0,
weakness = 'None', type = 'Unknown'):
    self.name = name
    self.hit_point = hp
    self.weakness = weakness
    self.type = type

  def get_type(self):
    return 'Main type: ' + self.type

  def get_move(self):
    return 'Basic move: ' + 'Quick Attack'

  def __str__(self):
    return "Name: " + self.name + ", HP: " +
str(self.hit_point) + ", Weakness: " + self.weakness

print('\n------------Basic Info:--------------')
pk = PokemonBasic()
print(pk)
print(pk.get_type())
print(pk.get_move())

print('\n------------Pokemon 1 Info:-------------')
charmander = PokemonExtra('Charmander', 39, 'Water',
'Fire')
print(charmander)
print(charmander.get_type())
print(charmander.get_move())

print('\n------------Pokemon 2 Info:-------------')
charizard = PokemonExtra('Charizard', 78, 'Water',
'Fire', 'Flying', ('Fire Spin', 'Fire Blaze'))
print(charizard)
print(charizard.get_type())
print(charizard.get_move())
```

OUTPUT:

```
------------Basic Info:--------------
Name: Default, HP: 0, Weakness: None
Main type: Unknown
Basic move: Quick Attack

------------Pokemon 1 Info:--------------
Name: Charmander, HP: 39, Weakness: Water
Main type: Fire
Basic move: Quick Attack

------------Pokemon 2 Info:--------------
Name: Charizard, HP: 78, Weakness: Water
Main type: Fire, Secondary type: Flying
Basic move: Quick Attack
Other move: Fire Spin, Fire Blaze
```

Task – 8

Implement the design of the **FootBallTeam** and the **CricketTeam** classes that inherit from **Team** class so that the following code generates the output below:

Driver Code

```
class Team:

    def __init__(self, name):
        self.name = "default"
        self.total_player = 5
    def info(self):
        print("We love sports")
# Write your code here.

class Team_test:
    def check(self, tm):
        print("=========================")
        print("Total Player: ", tm.total_player)
        tm.info()

f = FootBallTeam("Brazil")
c = CricketTeam("Bangladesh")
test = Team_test()
test.check(f)
test.check(c)
```

Output

```
=========================
Total Player: 11
Our name is Brazil
We play Football
We love sports
=========================
Total Player: 11
Our name is Bangladesh
We play Cricket
We love sports
```

Task – 9

Implement the design of the **Pikachu** and **Charmander** classes that are derived from the **Pokemon** class so that the following output is produced:

Driver Code

```python
class Pokemon:

    def __init__(self, p):
        self.pokemon = p
        self.pokemon_type = "Needs to be set"
        self.pokemon_weakness = "Needs to be set"
    def kind(self):
        return self.pokemon_type
    def weakness(self):
        return self.pokemon_weakness
    def what_am_i(self):
        print("I am a Pokemon.")

pk1 = Pikachu()
print("Pokemon:", pk1.pokemon)
print("Type:", pk1.kind())
print("Weakness:", pk1.weakness())
pk1.what_am_i()
print("========================")
c1 = Charmander()
print("Pokemon:", c1.pokemon)
print("Type:", c1.kind())
print("Weakness:", c1.weakness())
c1.what_am_i()
```

Output

```
Pokemon: Pikachu
Type: Electric
Weakness: Ground
I am a Pokemon.
I am Pikachu.
========================
Pokemon: Charmander
Type: Fire
Weakness: Water, Ground and Rock
I am a Pokemon.
I am Charmander.
```

Task – 10

Implement the design of the **CSE** and **EEE** classes that are derived from the Department class so that the following output is produced:

Driver Code

```
class Department:
    def __init__(self, s):
        self.semester = s
        self.name = "Default"
        self.id = -1

    def student_info(self):
        print("Name:", self.name)
        print("ID:", self.id)

    def courses(self, c1, c2, c3):
        print("No courses Approved yet!")


s1 = CSE("Rahim", 16101328,"Spring2016")
s1.student_info()
s1.courses("CSE110", "MAT110", "ENG101")
print("==================")
s2 = EEE("Tanzim", 18101326, "Spring2018")
s2.student_info()
s2.courses("Mat110", "PHY111", "ENG101")
print("==================")
s3 = CSE("Rudana", 18101326, "Fall2017")
s3.student_info()
s3.courses("CSE111", "PHY101", "MAT120")
print("==================")
s4 = EEE("Zainab", 19201623, "Summer2019")
s4.student_info()
s4.courses("EEE201", "PHY112", "MAT120")
```

Output

```
Name: Rahim
ID: 16101328
Courses Approved to this CSE student in
Spring2016 semester :
CSE110
MAT110
ENG101
==================
Name: Tanzim
ID: 18101326
Courses Approved to this EEE student in
Spring2018 semester:
Mat110
PHY111
ENG101
==================
Name: Rudana
ID: 18101326
Courses Approved to this CSE student in
Fall2017 semester:
CSE111
PHY101
MAT120
==================
Name: Zainab
ID: 19201623
Courses Approved to this EEE student in
Summer2019 semester:
EEE201
PHY112
MAT120
```

Task – 11

```python
class A:
    def __init__(self):
        self.temp = 4
        self.sum = 1
        self.y = 2
        self.y = self.temp - 2
        self.sum = self.temp + 3
        self.temp -= 2
    def methodA(self, m,  n):
        x = 0
        self.y = self.y + m + self.temp
        self.temp += 1
        x = x + 2 + n
        self.sum = self.sum + x + self.y
        print(x, self.y, self.sum)

class B(A):
    def __init__(self, b=None):
        super().__init__()
        self.x = 1
        self.sum = 2
        if b == None:
            self.y = self.temp + 3
            self.sum = 3 + self.temp + 2
            self.temp -= 1
        else:
            self.sum = b.sum
            self.x = b.x
    def methodB(self, m,  n):
        y = 0
        y = y + self.y
        self.x = y + 2 + self.temp
        self.methodA(self.x, y)
        self.sum = self.x + y + self.sum
        print(self.x, y, self.sum)
```

Write the output of the following code:

```python
a1 = A()
b1 = B()
b2 = B(b1)
a1.methodA(1, 1)
b1.methodA(1, 2)
b2.methodB(3, 2)
```

Output:

x	y	sum

Task – 12

```python
class A:
    temp = 4
    def __init__(self):
        self.sum = 0
        self.y = 0
        self.y = A.temp - 2
        self.sum = A.temp + 1
        A.temp -= 2
    def methodA(self, m,  n):
        x = 0
        self.y = self.y + m + (A.temp)
        A.temp += 1
        x = x + 1 + n
        self.sum = self.sum + x + self.y
        print(x, self.y, self.sum)

class B(A):
    x = 0
    def __init__(self,b=None):
        super().__init__()
        self.sum = 0
        if b==None:
            self.y = A.temp + 3
            self.sum = 3 + A.temp + 2
            A.temp -= 2
        else:
            self.sum = b.sum
            B.x = b.x
            b.methodB(2, 3)
    def methodB(self, m,  n):
        y = 0
        y = y + self.y
        B.x = self.y + 2 + A.temp
        self.methodA(B.x, y)
        self.sum = B.x + y + self.sum
        print(B.x, y, self.sum)
```

Write the output of the following code:

```
a1 = A()
b1 = B()
b2 = B(b1)
b1.methodA(1, 2)
b2.methodB(3, 2)
```

Output:

x	y	sum

Task – 13

```python
class A:
  temp = 3
  def __init__(self):
    self.sum = 0
    self.y = 0
    self.y = A.temp - 1
    self.sum = A.temp + 2
    A.temp -= 2

  def methodA(self, m, n):
    x = 0
    n[0] += 1
    self.y = self.y + m + A.temp
    A.temp += 1
    x = x + 2 + n[0]
    n[0] = self.sum + 2
    print(f"{x} {self.y} {self.sum}")

class B(A):
  x = 1
  def __init__(self, b = None):
    super().__init__()
    self.sum = 2
    if b == None:
      self.y = self.temp + 1
      B.x = 3 + A.temp + self.x
      A.temp -= 2
    else:
      self.sum = self.sum + self.sum
      B.x = b.x + self.x
  def methodB(self, m, n):
    y = [0]
    self.y = y[0] + self.y + m
    B.x = self.y + 2 +  self.temp - n
    self.methodA(self.x, y)
    self.sum = self.x + y[0] + self.sum
    print(f"{self.x} {y[0]} {self.sum}")
```

Write the output of the following code:

```python
x = [23]
a1 = A()
b1 = B()
b2 = B(b1)
a1.methodA(1, x)
b2.methodB(3, 2)
a1.methodA(1, x)
```

Output:

x	y	sum

Task – 14

```python
class A:
  temp = 7
  def __init__(self):
    self.sum, self.y = 0, 0
    self.y = A.temp - 1
    self.sum = A.temp + 2
    A.temp -= 3
  def methodA(self, m, n):
    x = 4
    n[0] += 1
    self.y = self.y + m + A.temp
    A.temp += 2
    x = x + 3 + n[0]
    n[0] = self.sum + 2
    print(f"{x} {self.y} {self.sum}")
  def get_A_sum(self):
    return self.sum
  def update_A_y(self, val):
    self.y = val
class B(A):
  x = 2
  def __init__(self, b = None):
    super().__init__()
    self.sum = 2
    if b == None:
      self.y = self.temp + 1
      B.x = 4 + A.temp + self.x
      A.temp -= 2
    else:
      self.sum = self.sum + self.get_A_sum()
      B.x = b.x + self.x
  def methodB(self, m, n):
    y = [0]
    self.update_A_y(y[0] + self.y + m)
    B.x = self.y + 4 +  self.temp - n
    self.methodA(self.x, y)
    self.sum = self.x + y[0] + self.get_A_sum()
    print(f"{self.x} {y[0]} {self.sum}")
```

Write the output of the following code:

```python
x = [32]
a1 = A()
b1 = B()
b2 = B(b1)
a1.methodA(2, x)
b2.methodB(2, 3)
a1.methodA(3, x)
```

Output:		
x	y	sum

https://bout.eveneer.xyz/evaluation-form