Task- 01

In [ ]:

```
1
2  class RealNumber:
3
4      def __init__(self, r=0):
5          self.__realValue = r
6      def getRealValue(self):
7          return self.__realValue
8      def setRealValue(self, r):
9          self.__realValue = r
10     def __str__(self):
11         return 'RealPart: '+str(self.getRealValue())
12
13 class ComplexNumber(RealNumber):
14     def __init__(self, r = 1, ip = 1):
15         super().__init__(float(r))
16         self.__ip = float(ip)
17     def getImagineValue(self):
18         return self.__ip
19     def __str__(self):
20         return f"{super().__str__() } \nImaginaryPart: {str(self.getImagineV
21
22 cn1 = ComplexNumber()
23 print(cn1)
24 print('---------')
25 cn2 = ComplexNumber(5,7)
26 print(cn2)
```

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

Task- 02

In [1]:

```
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)

class ComplexNumber(RealNumber):
    def __init__(self, number, imn):
        super().__init__(number)
        self.imn = imn
    def __str__(self):
        return "{} + {}i".format(super().__str__(), str(self.imn))
    def __add__(self, complexnumber):
        return "{} + {}i".format(super().__add__(complexnumber.number), (sel
    def __sub__(self, complexnumber):
        if complexnumber.imn > self.imn:
            return "{} - {}i".format(super().__sub__(complexnumber.number),
        else:
            return "{} + {}i".format(super().__sub__(complexnumber.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)
```

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

Task- 03

In [ ]:

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

    def getBalance(self):
        return self._balance

class CheckingAccount(Account):
    numberOfAccount = 0
    def __init__(self, balance = 0.0):
        super().__init__(balance)
        CheckingAccount.numberOfAccount += 1
    def __str__(self):
        if self.getBalance() != 0:
            return f"Account Balance: {self.getBalance():.2f}"
        else:
            return "Account Balance:{}".format(self.getBalance())

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

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

Task- 04

In [ ]:

```
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)

class Mango(Fruit):
    def __init__(self, formalin = True, name='Mango'):
        super().__init__(formalin, name)
    def __str__(self):
        if self.hasFormalin() == True:
            return "{}s are bad for you".format(self.getName())
        else:
            return "{}s are good for you".format(self.getName())

class Jackfruit(Fruit):
    def __init__(self, formalin = False, name='Jackfruit'):
        super().__init__(formalin, name)
    def __str__(self):
        if self.hasFormalin() == True:
            return "{}s are bad for you".format(self.getName())
        else:
```

```
            return "{}s are good for you".format(self.getName())

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

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

Task- 05

In [ ]:

```
1
2  class Exam:
3      def __init__(self,marks):
4          self.marks = marks
5          self.time = 60
6      def examSyllabus(self):
7          return "Maths , English"
8      def examParts(self):
9          return "Part 1 - Maths\nPart 2 - English\n"
10
11 class ScienceExam(Exam):
12     def __init__(self, marks, time, *nop):
13         super().__init__(marks)
14         self.time = time
15         self.nop = nop
16     def examSyllabus(self):
17         course = super().examSyllabus()
18         for k in self.nop:
19              course += " , " + k
20         return course
21     def examParts(self):
22         course = super().examParts()
23         for k in self.nop:
24             if self.nop.index(k) == len(self.nop):
25                 course += "Part {} - {}".format((self.nop.index(k)+3), k)
26             else:
27                 course += "Part {} - {}\n".format((self.nop.index(k)+3), k)
28         return course
29     def __str__(self):
30         return "Marks: {} Time: {} minutes Number of Parts: {}".format(self.
31
32
33 engineering = ScienceExam(100,90,"Physics","HigherMaths")
34 print(engineering)
35 print('---------------------------------')
36 print(engineering.examSyllabus())
37 print(engineering.examParts())
38 print('=================================')
```

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

```
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- 06

In [ ]:

```
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)

class Sphere(Shape3D):
    def __init__(self, name = 'Default', radius = 0):
        super().__init__(name, radius)
        print("Shape name: {} Area Formula: 4 * pi * r * r".format(self._nam
    def calc_surface_area(self):
        self._area = 2 * super().calc_surface_area() * self._radius
        return self._area
    def __str__(self):
        return "{}, Height: {}\nArea: {}".format(super().__str__(), self._he

class Cylinder(Shape3D):
    def __init__(self, name = 'Default', radius = 0, height = 0):
        super().__init__(name, radius)
        self._height = height
        print("Shape name: {} Area Formula: 2 * pi * r * (r + h)".format(sel
    def calc_surface_area(self):
        self._area = super().calc_surface_area() * (self._radius+self._heigh
        return self._area
    def __str__(self):
        return "{}, Height: {}\nArea: {}".format(super().__str__(), self._he


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

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

```
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- 07

In [ ]:

```
class PokemonBasic:
    def __init__(self, name = 'Default', hp = 0, weakness = 'None', type = 'I
        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) + ", Wea

class PokemonExtra(PokemonBasic):
    def __init__(self, name = 'Default', hp = 0, weakness = 'None', type = 'I
        super().__init__(name, hp, weakness, type)
        self.st = st
        self.other_move = other_move
    def get_type(self):
        if self.st == None:
            return super().get_type()
        else:
            return "{}  Secondary type: {}".format(super().get_type(), self.s
    def get_move(self):
        if self.other_move == None:
            return super().get_move()
        else:
            self.moves = []
            x = ""
            for move in self.other_move:
                self.moves.append(self.other_move)
                if len(self.other_move) == 1:
                    x = x + self.other_move[0]
                else:
                    for k in range((len(self.other_move)) - 1):
                        x = x + self.other_move[k] + ', '
                    x = x + self.other_move[-1]
```

```
39
40                  return "{}\nOther move: {}".format(super().get_move(), x)
41
42  print('\n------------Basic Info:--------------')
43  pk = PokemonBasic()
44  print(pk)
45  print(pk.get_type())
46  print(pk.get_move())
47  print('\n------------Pokemon 1 Info:-------------')
48  charmander = PokemonExtra('Charmander', 39, 'Water', 'Fire')
49  print(charmander)
50  print(charmander.get_type())
51  print(charmander.get_move())
52  print('\n------------Pokemon 2 Info:-------------')
53  charizard = PokemonExtra('Charizard', 78, 'Water', 'Fire', 'Flying', ('Fire S
54  print(charizard)
55  print(charizard.get_type())
56  print(charizard.get_move())
```

```
------------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- 08

In [ ]:

```
class Team:
    def __init__(self, name):
        self.name = "default"
        self.total_player = 5
    def info(self):
        print("We love sports")

class FootBallTeam(Team):
    def __init__(self, name = "default"):
        self.name = name
        self.total_player = 11
    def info(self):
        print("Our name is {}".format(self.name))
        if self.name == "Brazil":
            print("We  play Football")
        else:
            pass
        super().info()

class CricketTeam(Team):
    def __init__(self, name = "default"):
        self.name = name
        self.total_player = 11
    def info(self):
        print("Our name is {}".format(self.name))
        if self.name == "Bangladesh":
            print("We  play Cricket")
        else:
            pass
        super().info()

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)
```

```
========================
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- 09

In [ ]:

```
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.")

class Pikachu(Pokemon):
    def __init__(self, p = "Pikachu"):
        super().__init__(p)
    def kind(self):
        self.pokemon_type = "Electric"
        return self.pokemon_type
    def weakness(self):
        self.pokemon_weakness = "Ground"
        return self.pokemon_weakness
    def what_am_i(self):
        super().what_am_i()
        print("I am {}.".format(self.pokemon))

class Charmander(Pokemon):
    def __init__(self, p = "Charmander"):
        super().__init__(p)
    def kind(self):
        self.pokemon_type = "Fire"
        return self.pokemon_type
    def weakness(self):
        self.pokemon_weakness = "Water, Ground and Rock"
        return self.pokemon_weakness
    def what_am_i(self):
        super().what_am_i()
        print("I am {}.".format(self.pokemon))
```

```
39
40 pk1 = Pikachu()
41 print("Pokemon:", pk1.pokemon)
42 print("Type:", pk1.kind())
43 print("Weakness:", pk1.weakness())
44 pk1.what_am_i()
45 print("========================")
46 c1 = Charmander()
47 print("Pokemon:", c1.pokemon)
48 print("Type:", c1.kind())
49 print("Weakness:", c1.weakness())
50 c1.what_am_i()
```

```
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

In [ ]:

```
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!")

class CSE(Department):
    def __init__(self, name, ID, s):
        super().__init__(s)
        self.name = name
        self.id = ID
    def courses(self, *c):
        print("Courses Approved to this CSE student in {} semester:".format(s
        for k in c:
            print(k)

class EEE(Department):
    def __init__(self, name, ID, s):
        super().__init__(s)
        self.name = name
        self.id = ID
    def courses(self, *c):
        print("Courses Approved to this EEE student in {} semester:".format(s
        for k in c:
            print(k)

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")
```

```
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
```

In [ ]:

```
1
```