Question 1

In [1]:
```
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())

class ComplexNumber(RealNumber):
    def __init__(self,r=1,i=1):
        super().__init__(r)
        self.__realValue = r
        self.i = i

    def __str__(self):
        return f"RealPart: {float(self.getRealValue())}\nImaginaryPart: {float(self.i)}"

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

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

Question 2

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, r,i):
        self.realPart = int(str(r))
        self.imaginaryPart=i
    def __add__(self, other):
        r=self.realPart+other.realPart
        i=self.imaginaryPart+other.imaginaryPart
        return ComplexNumber(r,i)
    def __sub__(self, other):
        r=self.realPart-other.realPart
        i=self.imaginaryPart-other.imaginaryPart
        return ComplexNumber(r,i)
    def __str__(self):
        if self.imaginaryPart>0:
            return str(self.realPart)+" + "+str(self.imaginaryPart)+"i"
        else:
             return str(self.realPart)+" - "+str(abs(self.imaginaryPart))+"i"


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
```

Question 3

In [2]:
```
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):
        return f'Account Balance: {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)
```

```
Number of Checking Accounts:  0
Account Balance: 0.0
Account Balance: 100.0
Account Balance: 200.0
Number of Checking Accounts:  3
```

Question 4

In [3]:
```
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,testFruit):

    def __init__(self,formalin=True,name='Mango'):
        super().__init__(formalin,name)

    def __str__(self):
        return f'{self.name} are bad for you'

class Jackfruit(Fruit,testFruit):

    def __init__(self,formalin=False,name='JackFruit'):
        super().__init__(formalin,name)

    def __str__(self):
        return f'{self.name} are good for you'


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

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

Question 5

In [4]:
```
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"

class ScienceExam(Exam):

    def __init__(self,marks,time,*sub):
        super().__init__(marks)
        self.time = time
        self.sub = sub
        self.s = ''
        for i in self.sub:
            self.s = self.s + i + ' , '

        self.part = 2 + len(sub)

    def __str__(self):

        return f"Marks: {self.marks} Time: {self.time} minutes Number of Parts: {self.part}"

    def examSyllabus(self):
        return f"Maths , English , {self.s[:-2]}"

    def examParts(self):
        if len(self.sub)==2:
            return f"Part 1 - Maths\nPart 2 - English\nPart 3 - {self.s[:7]}\nPart 4 - {self.s[10:21]}"
        if len(self.sub)==3:
            return f"Part 1 - Maths\nPart 2 - English\nPart 3 - {self.s[:7]}\nPart 4 - {self.s[10:21]}\nPart 5 -

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())
```

```
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
```

Question 6

In [6]:
```
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: "+str(self._name)+','+" Area Formula: 4 * pi * r * r ")

    def calc_surface_area(self):
        self.area = 4 * Shape3D.pi * self._radius * self._radius

    def __str__(self):
        return f"Radius: {self._radius}, Height: {self._height}\nArea: {self.area}"


class Cylinder(Shape3D):

    def __init__(self, name = 'Default', radius = 0, height = 0):
        super().__init__(name,radius)
        self.height = height
        print("Shape name: "+str(self._name)+','+" Area Formula: 2 * pi * r * (r + h) ")

    def calc_surface_area(self):
        self.area = 2 * Shape3D.pi * self._radius * (self._radius+self.height)

    def __str__(self):
        return f"Radius: {self._radius}, Height: {self.height}\nArea: {self.area}"


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
```

Question 7

In [2]:
```
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

class PokemonExtra(PokemonBasic):
    def __init__(self,name = 'Default', hp = 0,weakness = 'None',*type ):
        super().__init__(name,hp,weakness,type)
        self.type = type
        self.s = ''
        for i in self.type:
            self.s = self.s + str(i) + ','
        self.s1 = self.s[:-1]
        self.s1 = self.s1.split(',')
    def get_type(self):
        if len(self.s1) >=2:
            return 'Main type: ' + self.s1[0] + ', Secondary type: ' + self.s1[1]
        else:
            return 'Main type: ' + self.s1[0]
    def get_move(self):
        if len(self.s1) >= 2:
            return 'Basic move: ' + 'Quick Attack'+"\nOther move: " + self.s1[2][2:-1]+', '+self.s1[3][2:-2]
        else:
            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())
```

```
------------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
```

In [ ]: