Ejercicios de Programación Orientada a Objetos

Nota: Los ejercicios son graduales, es decir, en cada ejercicio se van implementando mejoras a una (o varias) clase(s).

Ejercicio 1

Crear una nueva clase llamada **NumeroComplejo**. Esta clase tiene un atributo **x** para la coordenada en x e **y** para la coordenada en y. Representar un número complejo de la forma (x, y).

```
class NumeroComplejo:
  def __init__(self, x, y):
    self.x = x
    self.y = y

nX = int(input("Ingrese un numero para la variable x : "))
nY = int(input("Ingrese un numero para la variable y : "))

numeroComplejo1 = NumeroComplejo(nX, nY)
print(f'Número complejo : ({numeroComplejo1.x},{numeroComplejo1.y})')
```

```
Ingrese un numero para la variable x : 5
Ingrese un numero para la variable y : 4
Número complejo : (5,4)
```

Ejercicio 2

Definir para la clase **NumeroComplejo** un método que permita imprimir una instancia de la clase. Recordemos que al intentar imprimir un tipo definido por nosotros, se imprime la dirección de memoria

```
class NumeroComplejo:
  def __init__(self, x, y):
    self.__x = x
    self.__y = y

  @property
```

```
    def x(self):
      return self.__x

    @x.setter
    def x(self, x):
      self.__x = x

    @property
    def y(self):
      return self.__y

    @x.setter
    def y(self, y):
      self.__y = y

    def memoria(self):
      print(f'''
      Dirección de memoria de la variable x {id(self.__x)}
      Dirección de memoria de la variable y {id(self.__y)}
      ''')

    def imprimir(self):
      print(f'Número complejo : ({self.__x},{self.__y})')


nX = int(input("Ingrese un numero para la variable x : "))
nY = int(input("Ingrese un numero para la variable y : "))

numeroComplejo1 = NumeroComplejo(nX, nY)

numeroComplejo1.imprimir()
numeroComplejo1.memoria()
```

```
     Ingrese un numero para la variable x : 3
     Ingrese un numero para la variable y : 5
     Número complejo : (3,5)

         Dirección de memoria de la variable x 94029815781952
         Dirección de memoria de la variable y 94029815782016
```

Ejercicio 3

Definir la función **str** para la clase **NumeroComplejo** para poder imprimir usando la función print.

```
class NumeroComplejo:
  def __init__(self, x, y):
    self.__x = x
```

```
      self.__y = y

    @property
    def x(self):
      return self.__x

    @x.setter
    def x(self, x):
      self.__x = x

    @property
    def y(self):
      return self.__y

    @x.setter
    def y(self, y):
      self.__y = y

    def __str__(self):
      return 'Número complejo : (' + str(self.__x) + ',' + str(self.__y) + ')'

nX = int(input("Ingrese un numero para la variable x : "))
nY = int(input("Ingrese un numero para la variable y : "))

numeroComplejo1 = NumeroComplejo(nX, nY)
print(numeroComplejo1.__str__())
```

```
     Ingrese un numero para la variable x : 2
     Ingrese un numero para la variable y : 4
     Número complejo : (2,4)
```

Ejercicio 4

Definir una función que compara dos números complejos, ya que si dos objetos distintos tienen sus atributos iguales, no se consideran iguales.

```
class NumeroComplejo:
  def __init__(self, _x, _y):
    self.x = _x
    self.y = _y

  def imprimir(self):
    print("El valor de x es : " + str(self.x))
    print("El valor de y es : " + str(self.y))

  def comparar(self, c2):
    if self.x == c2.x and self.y == c2.y:
```

```
        return True
      return False

    def __str__(self):
      return 'Número complejo : (' + str(self.__x) + ',' + str(self.__y) + ')'

numeroComplejo1 = NumeroComplejo(4, 5)
numeroComplejo2 = NumeroComplejo(4, 5)

print(numeroComplejo1 == numeroComplejo2)
print(numeroComplejo1.comparar(numeroComplejo2))
```

```
False
True
```

Ejercicio 5

Implementar un método que sume dos numeros complejos sin modificiar los objetos originales, ya que se retorna un nuevo numero NumeroComplejo.

```
class NumeroComplejo:
  def __init__(self, x = 3, y = 6):
    self.__x = x
    self.__y = y

  @property
  def x(self):
    return self.__x

  @x.setter
  def x(self, x):
    self.__x = x

  @property
  def y(self):
    return self.__y

  @x.setter
  def y(self, y):
    self.__y = y

  def add(self, value):
    a = self.__x + value.x
    b = self.__y + value.y
    return NumeroComplejo(a,b)

  def __str__(self):
```

```
    return '(' + str(self.__x) + ',' + str(self.__y) + ')'


nX = int(input("Ingrese un numero para la variable x : "))
nY = int(input("Ingrese un numero para la variable y : "))

numeroComplejo1 = NumeroComplejo(nX, nY)
numeroComplejo2 = NumeroComplejo(nX, nY)
sumaTotal = numeroComplejo1.add(numeroComplejo2)

print(sumaTotal)
```

```
 Ingrese un numero para la variable x : 4
 Ingrese un numero para la variable y : 98
 (8,102)
```

Ahora, vamos a implementar un objeto **Bus**, que lleva a un grupo de pasajeros y además tiene un chofer. Los pasajeros y el chofer son todos personas, por lo que primero crearemos una clase **Persona** para representarlos.

Ejercicio 6

Crea una clase **Persona**. Sus atributos deben ser su nombre y su edad. Además crea un método cumpleaños, que aumente en 1 la edad de la persona.

```
class Persona:
  def __init__(self, nombre , edad):
    self.__nombre = nombre
    self.__edad = edad

  @property
  def nombre(self):
    return self.__nombre

  @nombre.setter
  def nombre(self, nombre):
    self.__nombre = nombre

  @property
  def edad(self):
    return self.__edad

  @edad.setter
  def edad(self, edad):
    self.__edad = edad
```

```
  def sumar_cumple(self):
    self.__edad+=1

  def __str__(self):
    return 'Nombre:' + str(self.__nombre) + ', Edad:' + str(self.__edad)

persona1 = Persona("Felipe", 27)
print(persona1)

persona1.sumar_cumple()
persona1.__str__()
```

```
Nombre:Felipe, Edad:27
'Nombre:Felipe, Edad:28'
```

Ejercicio 7

Para la clase anterior definir el método **str**. Debe retornar al menos el nombre de la persona.

+ Código + Texto

```
class Persona:
  def __init__(self, nombre , edad):
    self.__nombre = nombre
    self.__edad = edad

  @property
  def nombre(self):
    return self.__nombre

  @nombre.setter
  def nombre(self, nombre):
    self.__nombre = nombre

  @property
  def edad(self):
    return self.__edad

  @edad.setter
  def edad(self, edad):
    self.__edad = edad

  def __str__(self):
    return 'Nombre:' + str(self.__nombre) + ', Edad:' + str(self.__edad)

persona1 = Persona("Felipe", 27)
persona1.__str__()
```

Ejercicio 8

Extender la aplicación anterior con una clase **Bus**. Como atributo tiene un arreglo de pasajeros (inicialmente vacío), una capacidad (se ingresa en el constructor) y un chofer. Debes implementar el método ingresar_chofer(self, persona), que recibe una persona y queda como chofer del Bus si es mayor de 18 años. Cabe destacar que el chofer no se ingresa en el constructor.

```
class Persona:
  def __init__(self, nombre , edad):
    self.nombre = nombre
    self.edad = edad

  def __str__(self):
    return "Persona: " + self.nombre

class Bus:
  def __init__(self, capacidad):
    self.asientos = []
    self.capacidad = capacidad
    self.chofer = None

  def ingresar_chofer(self, persona):
    if persona.edad >= 18:
      self.chofer = persona
    else:
      print(persona.nombre + " no puede ser chofer")

persona1 = Persona("Felipe", 10)
persona2 = Persona("Enrrique", 22)

bus = Bus(10)
bus.ingresar_chofer(persona1)
bus.ingresar_chofer(persona2)
print(bus.chofer)
```

```
Felipe no puede ser chofer
Persona: Enrrique
```

Ejercicio 9

Extender la clase **Bus** con el método **subir_pasajero(self, persona)**. Este método sube a la persona al bus (i.e. La agrega al arreglo de asientos) siempre que el número de pasajeros en el Bus sea menor que la capcidad total.

```
class Persona:
  def __init__(self, nombre , edad):
    self.nombre = nombre
    self.edad = edad

  def __str__(self):
    return "Persona: " + self.nombre

class Bus:
  def __init__(self, capacidad):
    self.asientos = []
    self.capacidad = capacidad
    self.chofer = None

  def subir(self, persona):
    if self.capacidad > len(self.asientos):
      self.asientos.append(persona)
    else:
      print("No hay mas asientos en el bus, capacidad maxima " + str(self.capacidad) + " no p

persona1 = Persona("Felipe", 10)
persona2 = Persona("Enrrique", 22)
persona3 = Persona("Marco", 22)
persona4 = Persona("Juan", 22)

bus = Bus(3)
bus.subir(persona1)
bus.subir(persona2)
bus.subir(persona3)
bus.subir(persona4)

for pasajero in bus.asientos:
    print(pasajero)
```

```
No hay mas asientos en el bus, capacidad maxima 3 no puede subir Juan
Persona: Felipe
Persona: Enrrique
Persona: Marco
```

✓ 0 s completado a las 11:55