Creado por:

Isabel Maniega

Ejercicios

1) Crearemos un nuevo tipo llamado NumeroComplejo. Este tipo tiene un atributo x para la coordenada en x e y para la coordenada en y. Representa un número complejo de la forma (x, y).

In [1]:
```
class NumeroComplejo:
  # Definimos el constructor, que instancia los distintos objetos
  def __init__(self, x, y):
    # El primer atributo puede no llamarse self, pero se usa asi por convencion
    self.x = x
    self.y = y

nc = NumeroComplejo(1, 2)
print(nc.x) # Accedemos al valor de x para la instancia nc
```

```
1
```

2) Ahora defina dentro de la clase NumeroComplejo un función imprimir donde muestre los valores de x e y.

In [2]:
```
class NumeroComplejo:
    # Definimos el constructor, que instancia los distintos objetos
    def __init__(self, x, y):
        # El primer atributo puede no llamarse self, pero se usa asi por convencion
        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))

nc = NumeroComplejo(2,5)
nc.imprimir()
```

```
El valor de x es: 2
El valor de y es: 5
```

3) Define la función **str** para la clase NumeroComplejo para poder imprimir usando la función print.

In [3]:
```
class NumeroComplejo:
    # Definimos el constructor, que instancia los distintos objetos
    def __init__(self, x, y):
        # El primer atributo puede no llamarse self, pero se usa asi por convencion
        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 __str__(self):
        # Las funciones que parten y terminan con __ son funciones reservadas.
        # Existen varias además de __init__ y __str__
        s =  str(self.x) + "," + str(self.y)
        return s

nc = NumeroComplejo(6, 9)
print(nc)
```

```
6,9
```

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

In [4]:
```
class NumeroComplejo:
    # Definimos el constructor, que instancia los distintos objetos
    def __init__(self, x, y):
        # El primer atributo puede no llamarse self, pero se usa asi por convencion
        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 __str__(self):
        return "(" + str(self.x) + ", " + str(self.y) + ")"

    def son_iguales(self, c2):
        if self.x == c2.x and self.y == c2.y:
            return True
        return False

nc1 = NumeroComplejo(4, 5)
nc2 = NumeroComplejo(4, 5)
print(nc1 == nc2)
print(nc1.son_iguales(nc2))
```

```
False
True
```

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

In [5]:
```
class NumeroComplejo:
    # Definimos el constructor, que instancia los distintos objetos
    def __init__(self, x, y):
        # El primer atributo puede no llamarse self, pero se usa asi por convencion
        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 __str__(self):
        return "(" + str(self.x) + ", " + str(self.y) + ")"

    def son_iguales(self, c2):
        if self.x == c2.x and self.y == c2.y:
            return True
        return False
    def sumar_complejos(self, c2):
        a = self.x + c2.x
        b = self.y + c2.y
        return NumeroComplejo(a, b)
```

In [6]:
```
nc1 = NumeroComplejo(4, 5)
nc2 = NumeroComplejo(4, 5)
r = nc1.sumar_complejos(nc2)
print(r)
```

```
(8, 10)
```

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.

In [7]:
```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def cumpleanos(self):
        self.edad += 1

p = Persona("Jose", 10)
p.cumpleanos()
print(p.edad)
```

```
11
```

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

In [8]:
```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def cumpleanhos(self):
        self.edad += 1

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

p = Persona("Jose", 10)
print(p)
```

```
Persona: Jose
```

8) Extender la clase persona agregando un atributo saldo y un método transferencia(self, persona2, monto). El saldo representa el dinero que tiene la persona. El método transferencia hace que la Persona que llama el método le transfiera la cantidad monto al objeto persona2. Si no tiene el dinero suficiente no se ejecuta la acción.

In [9]:
```
class Persona:
    def __init__(self, nombre, edad, saldo):
        self.nombre = nombre
        self.edad = edad
        self.saldo = saldo

    def cumpleanos(self):
        self.edad += 1

    def transferencia(self, persona2, monto):
        if self.saldo >= monto:
            self.saldo -= monto
            persona2.saldo += monto
            print("Transferencia ok!")
        else:
            print("No se puede efectuar la transaccion")

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


p = Persona("Jose", 10, 2)
p2 = Persona("Pedro", 20, 20)
p2.transferencia(p, 7)
```

```
Transferencia ok!
```

Creado por:

Isabel Maniega