Alumno: ***Marcos Gómez Alonso***

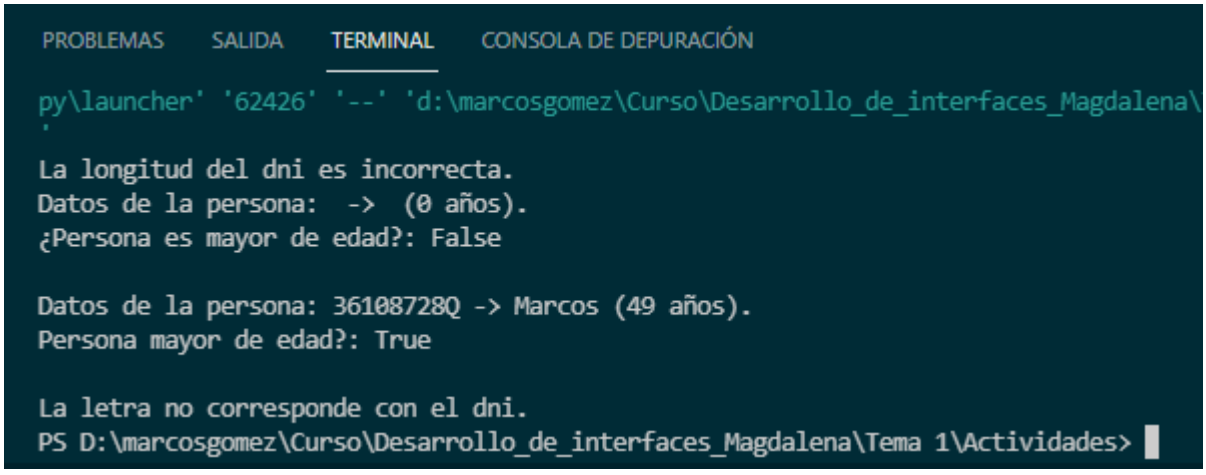
Ejercicio 1

'"Vamos a crear una clase llamada Persona. Sus atributos son: nombre, edad y DNI.
Construye los siguientes métodos para la clase:
• Un constructor, donde los datos pueden estar vacíos.
• Los setters y getters para cada uno de los atributos. Hay que validar las entradas de datos.
• mostrar(): Muestra los datos de la persona.
• esMayorDeEdad(): Devuelve un valor lógico indicando si es mayor de edad.'"

```
class Persona ():

    def __init__ (self, nombre = '', edad = 0, dni = ''):
        self.nombre = nombre
        self.edad = edad
        self.dni = dni

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

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

    @property
    def dni (self):
        return self.__dni

    @nombre.setter
    def nombre (self, nombre):
        if isinstance (nombre, str):
            self.__nombre = nombre
        else:
            print ('El dato introducido no es correcto.')

    @edad.setter
    def edad (self, edad):
        if isinstance (edad, int):
            if ((edad >= 0) and (edad <= 120)):
                self.__edad = edad
            else:
                print ('La edad no es correcta.')
        else:
            print ('El dato introducido no es correcto.')

    @dni.setter
    def dni (self, dni):
        if (len(dni) == 9):
```

```
        numero = int(dni [0:8])
        letra = dni [8:9]
        if ((isinstance (numero, int)) and (isinstance (letra, str))):

            #Obtener letra del dni
            letras = 'TRWAGMYFPDXBNJZSQVHLCKE'
            letra_correcta = letras [int (numero) % 23]
            if (letra == letra_correcta):
                self.__dni = dni
            else:
                self.__dni = ''
                print ('La letra no corresponde con el dni.')

        else:
            self.__dni = ''
            print ('El dato introducido no es correcto.')
    else:
        self.__dni = ''
        print ('La longitud del dni es incorrecta.')

def mostrar (self):
    return self.dni + " -> " + self.nombre + " (" + str (self.edad) + " años)."

def esMayorDeEdad (self):
    if (self.edad >= 18):
        return True
    else:
        return False
```

```
# Creación de la clase sin datos y funcionalidad de métodos.
yo = Persona ()
print ("Datos de la persona:", yo.mostrar())
print ("¿Persona es mayor de edad?:", yo.esMayorDeEdad(), "\n")

# Introducción de datos correctos.
yo.dni = "36108728Q"
yo.nombre = "Marcos"
yo.edad = 49
print ("Datos de la persona:", yo.mostrar())
print ("Persona mayor de edad?:", yo.esMayorDeEdad(), "\n")

# Introducción de un dni incorrecto.
# yo.dni = "36108728R"
yo.dni = "36108728R"
```



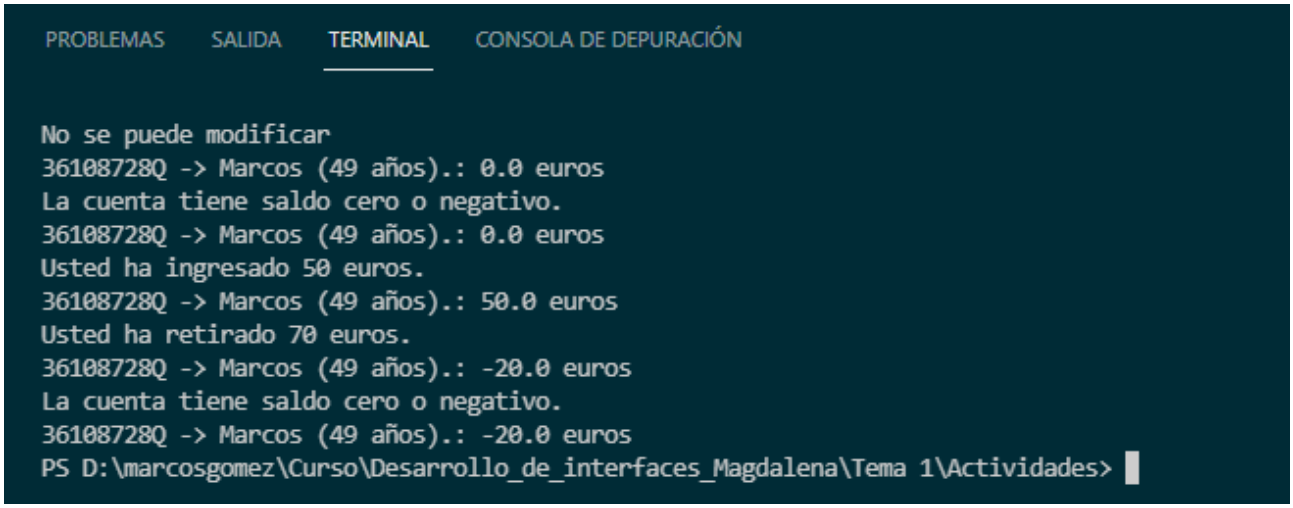
Ejercicio 2

'''Crea una clase llamada Cuenta que tendrá los siguientes atributos:
titular (que es una persona) y cantidad (puede tener decimales).
El titular será obligatorio y la cantidad es opcional.
Construye los siguientes métodos para la clase:
• Un constructor, donde los datos pueden estar vacíos.
• Los setters y getters para cada uno de los atributos.
  El atributo no se puede modificar directamente, solo ingresando o retirando dinero.
• mostrar(): Muestra los datos de la cuenta.
• ingresar(cantidad): se ingresa una cantidad a la cuenta,
  si la cantidad introducida es negativa, no se hará nada.
• retirar(cantidad): se retira una cantidad a la cuenta.
  La cuenta puede estar en números rojos.'''

```
from Ejercicios_POO_01_Persona import Persona

class Cuenta (Persona):

    def __init__ (self, nombre, edad, dni, cantidad = 0.0):
        super ().__init__ (nombre, edad, dni)
        self.cantidad = cantidad

    @property
    def titular (self):
        return super ().mostrar ()

    @property
    def cantidad (self):
        return self.__cantidad

    @titular.setter
    def __titular (self, nombre, edad, dni):
        super ().nombre = nombre
        super ().edad = edad
        super ().dni = dni

    @cantidad.setter
    def cantidad (self, cantidad):
        if (cantidad > 0):
            self.__cantidad = cantidad
        else:
            self.__cantidad = 0.0
            print ('No se puede modificar')

    def mostrar (self):
        return super ().mostrar () + ': ' + str (self.cantidad) + " euros"

    def ingresar (self, cantidad):
        if (cantidad >= 0):
            self.__cantidad += cantidad
            print ('Usted ha ingresado', cantidad, 'euros.')
```

```
        else:
            print ('No se ha ingresado ninguna cantidad.')

    def retirar (self, cantidad):
        if (cantidad >= 0):
            if (self.cantidad > 0):
                self.__cantidad -= cantidad
                print ('Usted ha retirado', cantidad, 'euros.')
            else:
                print ('La cuenta tiene saldo cero o negativo.')
        else:
            print ('No se ha retirado ninguna cantidad.')
```

```
# Introducir los datos del titular y la cantidad
miCuenta = Cuenta ('Marcos', 49, '36108728Q')
print (miCuenta.mostrar())

miCuenta.retirar (50)
print (miCuenta.mostrar())
miCuenta.ingresar (50)
print (miCuenta.mostrar())
miCuenta.retirar (70)
print (miCuenta.mostrar())
miCuenta.retirar (20)
print (miCuenta.mostrar())

miCuenta.titular ('Magdalena', 40, '36202101X')
```



Ejercicio 3

'''Vamos a definir ahora una “Cuenta Joven”,
para ello vamos a crear una nueva clase CuentaJoven
que deriva de la anterior. Cuando se crea esta nueva clase,
además del titular y la cantidad se debe guardar una bonificación
que estará expresada en tanto por ciento.
Construye los siguientes métodos para la clase:
• Un constructor.
• Los setters y getters para el nuevo atributo.
• En esta ocasión los titulares de este tipo de cuenta tienen que ser mayor de edad.,
por lo tanto hay que crear un método esTitularValido() que devuelve verdadero
si el titular es mayor de edad pero menor de 25 años y falso en caso contrario.
• Además la retirada de dinero sólo se podrá hacer si el titular es válido.
• El método mostrar() debe devolver el mensaje de “Cuenta Joven” y la bonificación de la cuenta.
Piensa los métodos heredados de la clase madre que hay que reescribir.'''

```
from Ejercicios_POO_02_Cuenta import Cuenta

class CuentaJoven (Cuenta):

    def __init__ (self, nombre, edad, dni, cantidad, bonificacion = 0):
        if (self.esTitularValido (edad)):
            super ().__init__ (nombre, edad, dni, cantidad)
            self.bonificacion = bonificacion
        else:
            print ('El titular debe tener entre 18 y 25 años.')

    @property
    def bonificacion (self):
        return self.__bonificacion

    @bonificacion.setter
    def bonificacion (self, bonificacion):
        self.__bonificacion = bonificacion

    def esTitularValido (self, edad):
        if ((edad >= 18) and (edad <= 25)):
            return True
        else:
            return False

    def mostrar (self):
        return 'Cuenta Joven: ' + super ().mostrar () + ' y bonificación del ' + str (self.bonificacion) +
'%.'

    def ingresar (self, cantidad):
        super ().ingresar (cantidad)

    def retirar (self, cantidad):
        if (self.esTitularValido (self.edad)):
            super ().retirar (cantidad)
```

```
        else:
            print ('La operación no se puede realizar.')
```

```
# Introducir los datos del titular y la cantidad
miCuentaJoven = CuentaJoven ('Marcos', 49, '36108728Q', 30, 5)
# print (miCuentaJoven.mostrar ())

miCuentaJoven = CuentaJoven ('Magdalena', 25, '36108728Q', 30, 13)
print (miCuentaJoven.mostrar ())
# miCuenta.cantidad = 75

miCuentaJoven.retirar (50)
print (miCuentaJoven.mostrar ())
miCuentaJoven.ingresar (50)
print (miCuentaJoven.mostrar ())
```