

**Realizar los siguientes ejercicios en lenguaje Python:****Ejercicio 1**

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.

(Edad >0, DNI Limite de 10 caracteres y opcional que valide que solo sean números)

- 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,dni):
```

```
    self.nombre= nombre
```

```
    self.edad= edad
```

```
    self.dni= dni
```

```
@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
```

```
    self.valEdad()
```

```
@property
```

```
def DNI(self):
```

```
    return self.dni
```

```
@DNI.setter
```

```
def DNI(self,dni):
```

```
    self.dni=dni
```

```
    self.valDNI()
```

```
def valDNI(self):
```

```
    tipo= type(1)
```

```
    if(type(self.dni)==tipo and len(str(self.dni))<=10):
```



```
        print("Tu ID es valido")
    else:
        print("Tu ID no es valido, revisalo e intenta de nuevo")
        self.dni= 0

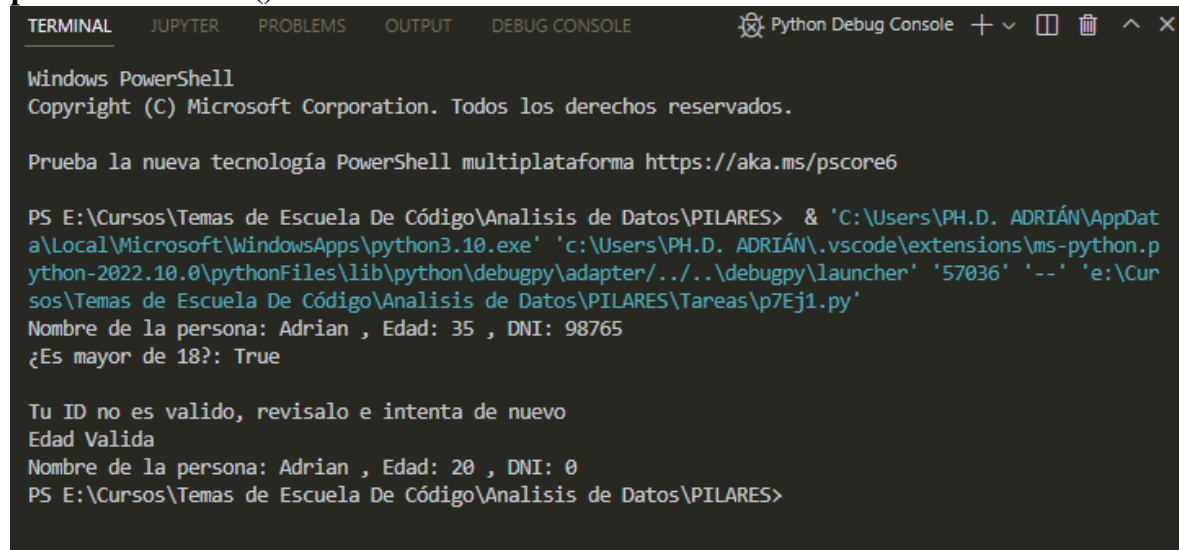
    def valEdad(self):
        if(self.edad>0):
            print("Edad Valida")
        else:
            print("Edad invalida")
            self.edad= 0

    def mostrar(self):
        print("Nombre de la persona:", self.nombre, ", Edad:",self.edad,"",
DNI:",self.dni)

    def esMayorDeEdad(self):
        Medad=self.edad>=18
        print("¿Es mayor de 18?:", Medad)

persona1=Persona("Adrian",35,98765)

persona1.mostrar()
persona1.esMayorDeEdad()
print()
persona1.DNI="4580uit"
persona1.Edad=20
persona1.mostrar()
```



```
TERMINAL  JUPYTER  PROBLEMS  OUTPUT  DEBUG CONSOLE  Python Debug Console  + v  [ ]  [X]  ^ x

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS E:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES> & 'C:\Users\PH.D. ADRIÁN\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\PH.D. ADRIÁN\.vscode\extensions\ms-python.python-2022.10.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '57036' '--' 'e:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES\Tareas\p7Ej1.py'
Nombre de la persona: Adrian , Edad: 35 , DNI: 98765
¿Es mayor de 18?: True

Tu ID no es valido, revisalo e intenta de nuevo
Edad Valida
Nombre de la persona: Adrian , Edad: 20 , DNI: 0
PS E:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES>
```



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, sólo 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.

class Cuenta:

```
def __init__(self, titular, cantidad=0):  
    self.__titular= titular  
    self.__cantidad= float(cantidad)
```

@property

```
def Titular(self):  
    return self.__titular
```

@Titular.setter

```
def Titular(self, titular):  
    self.__titular= titular
```

@property

```
def Cantidad(self):  
    return self.__cantidad
```

@Cantidad.setter

```
def Cantidad(self, cantidad=0):  
    self.__cantidad= float(cantidad)  
    self.valEdad()
```

def mostrar(self):

```
    print("Nombre:", self.__titular, ", Estado de cuenta:", self.__cantidad,)
```

def ingresar(self, cantidad):

```
    if cantidad>0:  
        self.__cantidad=self.__cantidad+float(cantidad)
```

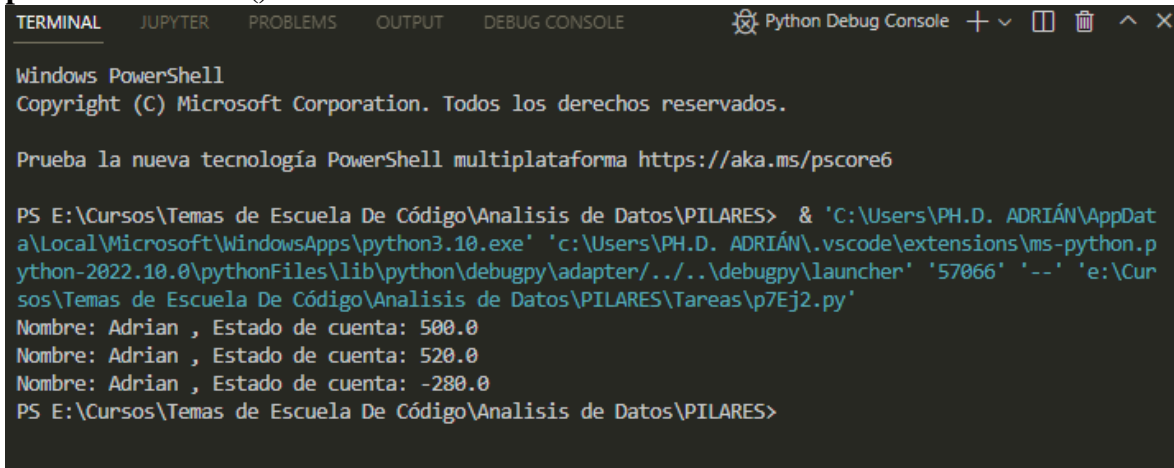
def retirar(self, cantidad):



```
if cantidad>0:  
    self.__cantidad=self.__cantidad-float(cantidad)
```

```
persona1=Cuenta("Adrian",500)
```

```
persona1.mostrar()  
persona1.ingresar(20)  
persona1.mostrar()  
persona1.retirar(800)  
persona1.mostrar()
```



```
TERMINAL JUPYTER PROBLEMS OUTPUT DEBUG CONSOLE Python Debug Console + - [] X  
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos los derechos reservados.  
  
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6  
  
PS E:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES> & 'C:\Users\PH.D. ADRIÁN\AppData  
a\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\PH.D. ADRIÁN\.vscode\extensions\ms-python.p  
ython-2022.10.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '57066' '--' 'e:\Cur  
sos\Temas de Escuela De Código\Análisis de Datos\PILARES\Tareas\p7Ej2.py'  
Nombre: Adrian , Estado de cuenta: 500.0  
Nombre: Adrian , Estado de cuenta: 520.0  
Nombre: Adrian , Estado de cuenta: -280.0  
PS E:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES>
```

Ejercicio 3

Vamos a definir ahora una “Cuenta Joven”, para ello vamos a crear una nueva clase Cuenta Joven 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 es Titular Válido () 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 superclase que hay que reescribir.

class Cuenta:

```
def __init__(self,titular, cantidad=0):  
  
    self.__titular= titular  
  
    self.__cantidad= float(cantidad)
```

**@property****def Titular(self):****return self.__titular****@Titular.setter****def Titular(self,titular):****self.__titular= titular****@property****def Cantidad(self):****return self.__cantidad****@Cantidad.setter****def Cantidad(self,cantidad=0):****self.__cantidad= float(cantidad)****self.valEdad()****def mostrar(self):****print("Nombre:", self.__titular, ", Estado de cuenta:",self.__cantidad,)****def ingresar(self,cantidad):****if cantidad>0:****self.__cantidad=self.__cantidad+float(cantidad)****def retirar(self,cantidad):****if cantidad>0:****self.__cantidad=self.__cantidad-float(cantidad)****class CuentaJoven(Cuenta):**



```
def __init__(self,titular,edad, cantidad=0, bonificacion=0):
```

```
    self.edad=edad
```

```
    Cuenta.__init__(self,titular,cantidad)
```

```
    self.bonificacion= bonificacion
```

```
@property
```

```
def Bono(self):
```

```
    return self.bonificacion
```

```
@Bono.setter
```

```
def Bono(self,bonificacion):
```

```
    self.bonificacion= bonificacion
```

```
    self.valEdad()
```

```
def TitularValido(self):
```

```
    Vedad=(self.edad>=18 and self.edad<25)
```

```
    print("¿Aplica para cuenta joven?:", Vedad)
```

```
    return Vedad
```

```
def mostrar(self):
```

```
    Vedad=CuentaJoven.TitularValido(self)
```

```
    if (Vedad==True):
```

```
        print("Cuenta Joven")
```

```
    else:
```

```
        print("Cuenta Normal")
```

```
        self.bonificacion=0
```

```
    Cuenta.mostrar(self)
```

```
    print("Con bonificacion de:",self.bonificacion,"%" )
```



```
def retirar(self,cantidad):
```

```
    Vedad=(self.edad>=18 and self.edad<25)
```

```
    if Vedad==True:
```

```
        Cuenta.retirar(self,cantidad)
```

```
        print("Operacion realizada con exito")
```

```
    else:
```

```
        print("Edad no valida para retirar")
```

```
persona2=CuentaJoven("Adrian",35,600,5)
```

```
persona2.mostrar()
```

```
print()
```

```
persona2.retirar(55)
```

```
persona2.mostrar()
```

```
print()
```

```
print()
```

```
persona1=CuentaJoven("Juan",22,300,5)
```

```
persona1.mostrar()
```

```
print()
```

```
persona1.retirar(20)
```

```
persona1.mostrar()
```



```
TERMINAL  JUPYTER  PROBLEMS  OUTPUT  DEBUG CONSOLE  Python Debug Console  + - □ □ ^ X

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS E:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES> & 'C:\Users\PH.D. ADRIÁN\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\PH.D. ADRIÁN\.vscode\extensions\ms-python.python-2022.10.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '57077' '--' 'e:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES\Tareas\p7Ej3.py'
¿Aplica para cuenta joven?: False
Cuenta Normal
Nombre: Adrian , Estado de cuenta: 600.0
Con bonificacion de: 0 %

Edad no valida para retirar
¿Aplica para cuenta joven?: False
Cuenta Normal
Nombre: Adrian , Estado de cuenta: 600.0
Con bonificacion de: 0 %

¿Aplica para cuenta joven?: True
Cuenta Joven
Nombre: Juan , Estado de cuenta: 300.0
Con bonificacion de: 5 %

Operacion realizada con exito
¿Aplica para cuenta joven?: True
Cuenta Joven
Nombre: Juan , Estado de cuenta: 280.0
Con bonificacion de: 5 %
PS E:\Cursos\Temas de Escuela De Código\Análisis de Datos\PILARES>
```