**AD: Práctica 7 POO.**

|  |  |
| --- | --- |
| **Nombre:** | **Roberto Jaime Rico Sandoval** |
| **Folio:** | **964NB09** |
| **Nombre del Pilares** | **Huipulco** |

**Objetivo: Comprobar que el alumno conoce y domina clases y objetos en Python.**

**Instrucciones: Resuelve cada ejercicio en python, pega el código correspondiente al ejercicio.**

Ejercicio 1 (2 puntos)

Vamos a crear una clase llamada Persona. Sus atributos son: nombre, edad y

DNI. Construye los siguientes métodos para la clase:

● Un constructor.

● Los setters y getters para cada uno de los atributos.

● mostrar(): Muestra los datos de la persona.

● esMayorDeEdad(): Devuelve un valor lógico indicando si es mayor de edad.

Respuesta:

"""

Autor: Roberto Jaime Rico Sandoval.

Fille: Ejercicio 1 / Práctica 6

Date: 12/ 06/ 2022

Folio: 964NB09

"""

class Personas():

    def \_\_init\_\_(self, nombre, edad, dni):

        self.nombre = nombre

        self.edad = edad

        self.dni = dni

    def setNombre(self, nombre):

        self.nombre = nombre

    def getNombre(self):

        return self.nombre

    def setEdad(self, edad):

        self.edad = edad

    def getEdad(self):

        return self.edad

    def setDni(self, dni):

        self.nombre = dni

    def getDni(self):

        return self.dni

    def mostrar(self):

        return print(f"\nLa persona se llama: {self.nombre}\nLa persona tiene: {self.edad}\nDNI de la persona: {self.dni}")

    def esMayorDeEdad(self):

        eva = True

        if self.edad >= 18:

            return print(f"\n{self.nombre}: ¿Es mayor de edad?  -  {eva}")

        else:

            return print(f"\n{self.nombre}: ¿Es mayor de edad?  -  {not(eva)}")

Persona1 = Personas("Roberto Sandoval", 12, 169024)

print(Persona1.mostrar())

print(Persona1.esMayorDeEdad())

Ejercicio 2 (2 puntos)

Crea una clase llamada Cuenta que tendrá los siguientes atributos: titular (que es

una persona) y cantidad (puede tener decimales). 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 pero no recibir una cantidad negativa.

Respuesta:

"""

Autor: Roberto Jaime Rico Sandoval.

Fille: Ejercicio 2 / Práctica 6

Date: 12/ 06/ 2022

Folio: 964NB09

"""

class Cuenta():

    def \_\_init\_\_(self, titular = "User", cantidad = 0.0, edad = 18):

        self.titular = titular

        self.\_cantidad = cantidad

        self.edad = edad

    def setTitular(self, titular):

        self.titular = titular

    def getTitular(self):

        return self.titular

    def setTitular(self, cantidad):

        self.\_cantidad = cantidad

    def getCantidad(self):

        return self.\_cantidad

    def setEdad(self, edad):

        self.edad = edad

    def getEdad(self):

        return self.edad

    def mostrar(self):

        return print(f"\nTitular: {self.titular}\nEstado de la cuenta: ${self.\_cantidad}")

    def ingresar(self):

        deposito = float(input("\n¿Cuánto dinero deseas depositar?  -  $"))

        if deposito > 0:

            self.\_cantidad += deposito

            return print(f"Estado de cuenta: ${self.\_cantidad}")

        elif deposito <= 0:

            pass

    def retirar(self):

        retiro = float(input("\n¿Cuánto dinero deseas retirar?  -  "))

        while retiro <= 0:

            print(f"\n\nDato erroneo: ${retiro}. Vuelve a intentarlo")

            retiro = float(input("\n¿Cuánto dinero deseas retirar?  -  $"))

        if retiro <= self.\_cantidad:

            return print(f"\nOperación satisfactoria.\nEstado de cuenta: ${(self.\_cantidad - retiro)}")

        elif retiro > self.\_cantidad:

            return print(f"\nOperación invalida. Saldo insuficiente.\nEstado de cuenta: ${self.\_cantidad}")

cuenta1 = Cuenta("Joaquín Gúzman", 2100)

cuenta1.mostrar()

cuenta1.ingresar()

cuenta1.retirar()

7.3 Ejercicio 3 (2 puntos)

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.

Respuesta:

"""

Autor: Roberto Jaime Rico Sandoval.

Fille: Ejercicio 2 / Práctica 6

Date: 12/ 06/ 2022

Folio: 964NB09

"""

import random

class Cuenta():

    def \_\_init\_\_(self, titular = "User", cantidad = 0.0):

        self.titular = titular

        self.\_cantidad = cantidad

    def setTitular(self, titular):

        self.titular = titular

    def getTitular(self):

        return self.titular

    def setTitular(self, cantidad):

        self.\_cantidad = cantidad

    def getCantidad(self):

        return self.\_cantidad

    def mostrar(self):

        return print(f"\nTitular: {self.titular}\nEstado de la cuenta: ${self.\_cantidad}")

    def ingresar(self):

        deposito = float(input("\n¿Cuánto dinero deseas depositar?  -  $"))

        if deposito > 0:

            self.\_cantidad += deposito

            return print(f"Estado de cuenta: ${self.\_cantidad}")

        elif deposito <= 0:

            pass

    def retirar(self):

        retiro = float(input("\n¿Cuánto dinero deseas retirar?  -  "))

        while retiro <= 0:

            print(f"\n\nDato erroneo: ${retiro}. Vuelve a intentarlo")

            retiro = float(input("\n¿Cuánto dinero deseas retirar?  -  $"))

        if retiro <= self.\_cantidad:

            return print(f"\nOperación satisfactoria.\nEstado de cuenta: ${(self.\_cantidad - retiro)}")

        elif retiro > self.\_cantidad:

            return print(f"\nOperación invalida. Saldo insuficiente.\nEstado de cuenta: ${self.\_cantidad}")

edad = int(input("\n\n\*\*\* Bienvenido al sistema de cuenta jóven \*\*\*\n¿Cuantos años tienes?  -  "))

class cuentaJoven(Cuenta):

    def \_\_init\_\_(self, titular="User", cantidad=0.0, bonificacion=0.2):

        Cuenta.\_\_init\_\_(self, titular, cantidad)

        self.bonificacion = bonificacion

    def setBonificacion(self, bonificacion):

        self.bonificacion = bonificacion

    def getBonificacion(self):

        return self.bonificacion

    def titularValido(self):

        eva = True

        if edad >= 18 and edad <= 25:

            print(f"\nEl titular de la cuenta es valido, es mayor de edad: {edad}\n{eva}")

            print(f"\n¿Deseas retirar dinero?\n1) sí\n2) No")

            re = int(input("\n\nRespuesta:  "))

            while re < 1 or re > 2:

                print(f"\nDato erroneo {re}. vuelve a intentarlo.")

                print(f"\n¿Deseas retirar dinero?\n1) sí\n2) No")

                re = int(input("\n\nRespuesta:  "))

            if re == 1:

                print(f"Estado de cuenta: ${estudiante.\_cantidad}")

                estudiante.retirar()

            elif re == 2:

                print(f"\nSaliendo del sistema, hasta luego {estudiante.titular}")

        else:

            print(f"El titular es invalido, es menor de edad o supera los 25 años: [{edad}\n{not(eva)}]")

    def mostrarCuentaJoven(self):

        if edad >= 18 and edad <= 25:

            self.bonificacion = random.uniform(0.02, 0.09)

            neto = (round(self.bonificacion,3) \* self.\_cantidad)

            total = (neto + self.\_cantidad)

            rentabilidad = round(estudiante.bonificacion \* 100,1)

            print(f"\nLa rentabilidad de la bonificación para la cuenta de {estudiante.titular} es de: {rentabilidad}%")

            print(f"\n \*\* Datos de la cuenta \*\*\n\nTitular: {self.titular}\nEstado de la cuenta bruto: ${self.\_cantidad}\nEstado de la cuenta neto: ${total}")

        else:

            print(f"\n \*\* Datos de la cuenta \*\*\n\nTitular: {self.titular}\nEstado de la cuenta bruto: ${self.\_cantidad}")

estudiante = cuentaJoven("Samuel Faret", 2500)

estudiante.titularValido()

estudiante.mostrarCuentaJoven()

**Repositorio de imágenes:** <https://drive.google.com/drive/folders/1sMRWvT_ArGLdS-XaF2zPBuByU4PE1RFY?usp=sharing>

**Repositorio de códigos:** <https://drive.google.com/drive/folders/1QmHqFEXTLyuIq_xtDww5cEV5qdNEXNlZ?usp=sharing>