**Trabajo práctico nro. 9**

|  |  |  |
| --- | --- | --- |
| Logo Departamento 2.jpg | **Asignatura: Programación I** | |
|  | |
| **Cursado:**Primer Trimestre | **Horas** **semanales**: 4 |
| **Carrera**: *Tecnicatura Universitaria en Programación* | **Nivel (Año):** |
| **Ciclo Lectivo: 2023** |

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* Person:

*def* \_\_init\_\_(*self*, *name*="", *age*=0, *dni*=""):

*self*.\_\_name = *name*

*self*.\_\_age = *age*

*self*.\_\_dni = *dni*

*def* get\_name(*self*):

return *self*.\_\_name

*def* set\_name(*self*, *name*):

*self*.\_\_name = *name*

*def* get\_age(*self*):

return *self*.\_\_age

*def* set\_age(*self*, *age*):

if *age* >= 0:

*self*.\_\_age = *age*

else:

print("La edad no puede ser un valor negativo.")

*def* get\_dni(*self*):

return *self*.\_\_dni

*def* set\_dni(*self*, *dni*):

if len(*dni*) == 8:

*self*.\_\_dni = *dni*

else:

print("El DNI debe tener 8 dígitos.")

*def* showPerson(*self*):

print(*f*"Nombre: {*self*.\_\_name}")

print(*f*"Edad: {*self*.\_\_age}")

print(*f*"DNI: {*self*.\_\_dni}")

*def* isAdult(*self*):

return *self*.\_\_age >= 18

# Ejemplo de uso:

persona1 = Person("Juan", 25, "12345678")

persona1.showPerson()

print("¿Es mayor de edad?", persona1.isAdult())

1. 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* Person:

*def* \_\_init\_\_(*self*, *name*, *age*, *dni*):

*self*.name = *name*

*self*.age = *age*

*self*.dni = *dni*

*def* \_\_str\_\_(*self*):

return *f*"Nombre: {*self*.name}, Edad: {*self*.age}, DNI: {*self*.dni}"

*class* Account:

*def* \_\_init\_\_(*self*, *owner*, *quantity*=0.0):

*self*.owner = *owner*

*self*.\_\_quantity = *quantity*

*def* get\_quantity(*self*):

return *self*.\_\_quantity

*def* showInfo(*self*):

print("Datos del titular de la cuenta:")

print(*self*.owner)

print(*f*"Saldo disponible: {*self*.\_\_quantity} euros")

*def* enter(*self*, *quantity*):

if *quantity* > 0:

*self*.\_\_quantity += *quantity*

*def* takeOut(*self*, *quantity*):

if *quantity* > 0:

*self*.\_\_quantity -= *quantity*

# Ejemplo de uso:

persona1 = Person("Juan", 25, "12345678")

cuenta1 = Account(persona1, 1000.0)

cuenta1.showInfo()

cuenta1.enter(500.0)

cuenta1.showInfo()

cuenta1.takeOut(200.0)

cuenta1.showInfo()

1. Desarrollar un programa que cargue los datos de un triángulo. Implementar una clase con los métodos para inicializar los atributos, imprimir el valor del lado con un tamaño mayor y el tipo de triángulo que es (equilátero, isósceles o escaleno).

*class* Triangle:

*def* \_\_init\_\_(*self*, *side1*, *side2*, *side3*):

*self*.side1 = *side1*

*self*.side2 = *side2*

*self*.side3 = *side3*

*def* type\_triangle(*self*):

if *self*.side1 == *self*.side2 == *self*.side3:

return "Equilátero"

elif *self*.side1 == *self*.side2 or *self*.side1 == *self*.side3 or *self*.side2 == *self*.side3:

return "Isósceles"

else:

return "Escaleno"

*def* obtain\_major\_side(*self*):

return max(*self*.side1, *self*.side2, *self*.side3)

# Función principal

*def* main():

side1 = float(input("Ingrese la longitud del primer lado del triángulo: "))

side2 = float(input("Ingrese la longitud del segundo lado del triángulo: "))

side3 = float(input("Ingrese la longitud del tercer lado del triángulo: "))

tri = Triangle(side1, side2, side3)

print(*f*"El tipo de triángulo es: {tri.type\_triangle()}")

print(*f*"El lado con mayor longitud es: {tri.obtain\_major\_side()}")

if \_\_name\_\_ == "\_\_main":

main()

1. Realizar una clase que administre una agenda. Se debe almacenar para cada contacto el nombre, el teléfono y el email. Además deberá mostrar un menú con las siguientes opciones:

* Añadir contacto
* Lista de contactos
* Buscar contacto
* Editar contacto
* Cerrar agenda

*class* Contact:

*def* \_\_init\_\_(*self*, *name*, *phone*, *email*):

*self*.name = *name*

*self*.phone = *phone*

*self*.email = *email*

*def* \_\_str\_\_(*self*):

return *f*"Nombre: {*self*.name}\nTeléfono: {*self*.phone}\nEmail: {*self*.email}"

*class* ContactList:

*def* \_\_init\_\_(*self*):

*self*.contacts = []

*def* add\_contacts(*self*, *name*, *phone*, *email*):

new\_contact = ContactList(*name*, *phone*, *email*)

*self*.contacts.append(new\_contact)

*def* list\_contacts(*self*):

if not *self*.contacts:

print("La agenda está vacía.")

else:

for i, contact in enumerate(*self*.contacts, *start*=1):

print(*f*"Contacto {i}:\n{contact}")

*def* search\_contact(*self*, *name*):

for contact in *self*.contacts:

if contact.name == *name*:

print("Contacto encontrado:\n", contact)

return

print(*f*"No se encontró ningún contacto con el nombre '{*name*}'.")

*def* update\_contact(*self*, *name*, *new\_phone*, *new\_email*):

for contact in *self*.contacts:

if contact.name == *name*:

contact.phone = *new\_phone*

contact.email = *new\_email*

print("Contacto actualizado.")

return

print(*f*"No se encontró ningún contacto con el nombre '{*name*}'.")

# Función principal

*def* main():

my\_contact\_list = ContactList()

while True:

print("\nMenú de la agenda:")

print("1. Añadir contacto")

print("2. Lista de contactos")

print("3. Buscar contacto")

print("4. Editar contacto")

print("5. Cerrar agenda")

option = input("Seleccione una opción (1/2/3/4/5): ")

if option == "1":

name = input("Nombre del contacto: ")

phone = input("Teléfono del contacto: ")

email = input("Email del contacto: ")

my\_contact\_list.add\_contacts(name, phone, email)

elif option == "2":

my\_contact\_list.list\_contacts()

elif option == "3":

name = input("Nombre del contacto a buscar: ")

my\_contact\_list.search\_contact(name)

elif option == "4":

name = input("Nombre del contacto a editar: ")

new\_phone = input("Nuevo teléfono: ")

new\_email = input("Nuevo email: ")

my\_contact\_list.update\_contact(name, new\_phone, new\_email)

elif option == "5":

print("Agenda cerrada. ¡Hasta luego!")

break

else:

print("Opción no válida. Por favor, seleccione una opción válida.")

if \_\_name\_\_ == "\_\_main\_\_":

main()