**INFORME DE TRABAJO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INFORMACIÓN BÁSICA** | | | | | |
| **ASIGNATURA:** | Programación Web 2 | | | | |
| **TÍTULO DE LA PRÁCTICA:** | *POO en Python* | | | | |
| **NÚMERO DE PRÁCTICA:** | *--* | **AÑO LECTIVO:** | *2024A* | **NRO. SEMESTRE:** | *III* |
| **FECHA DE PRESENTACIÓN** | *30/05/2024* | **HORA DE PRESENTACIÓN** | *10:00* | | |
| **INTEGRANTE (s):**  - Yenaro Noa Camino Joel  - Choquehuanca Berna William Herderson | | | | **NOTA:** |  |
| **DOCENTE(s):**  *MARCELA QUISPE CRUZ* | | | | | |

|  |
| --- |
| **SOLUCIÓN Y RESULTADOS** |
| 1. **SOLUCIÓN DE EJERCICIOS/PROBLEMAS**   *Ejercicio1:*  *import* json  class Contacto:      def \_\_init\_\_(*self*, *name*: str, *number*: str, *direction*: str, *relation*: str) -> None:  *self*.name = *name*  *self*.number = *number*  *self*.direction = *direction*  *self*.relation = *relation*        def \_\_json\_\_(*self*):  *return* {              'name': *self*.name,              'number': *self*.number,              'direction': *self*.direction,              'relation': *self*.relation          }  class Agenda:      def \_\_init\_\_(*self*):  *self*.contactos = []  *self*.path = "pw2Teo/tarea3/data.txt"      def search(*self*, *name*: str):  *for* i, contacto *in* enumerate(*self*.contactos):  *if* contacto.name == *name*:  *return* i, contacto  *return* None, None        def add(*self*, *contacto*):          index, \_ = *self*.search(*contacto*.name)  *if* index is not None:  *self*.edit(index, *contacto*)  *else*:  *self*.contactos.append(*contacto*)        def edit(*self*, *i*, *contacto*):  *self*.contactos[*i*] = *contacto*      def remove(*self*, *name*: str):          index, \_ = *self*.search(*name*)  *if* index is not None:  *del* *self*.contactos[index]              print(f"Contacto {*name*} eliminado.")  *else*:              print(f"Contacto {*name*} no encontrado.")      def \_\_str\_\_(*self*):          str\_result = ""  *for* i, contacto *in* enumerate(*self*.contactos):              str\_result += f"\nNombre: {contacto.name}, Numero: {contacto.number}, Dirección: {contacto.direction}, Relación: {contacto.relation}"  *return* str\_result      def guardar(*self*):          contactos\_json = [c.\_\_json\_\_() *for* c *in* *self*.contactos]          json\_string = json.dumps(contactos\_json)  *with* open(*self*.path, "w") *as* file:              file.write(json\_string)          print("Contactos guardados correctamente.")        def recuperar(*self*):  *try*:  *with* open(*self*.path, "r") *as* file:                  json\_string = file.read()              contactos\_json = json.loads(json\_string)  *self*.contactos = [Contacto(c["name"], c["number"], c["direction"], c["relation"]) *for* c *in* contactos\_json]              print("Contactos recuperados correctamente.")  *except* FileNotFoundError:              print("Archivo no encontrado. No se pudieron recuperar los contactos.")  *except* json.JSONDecodeError:              print("Error al decodificar el archivo JSON. No se pudieron recuperar los contactos.")  class Principal:      def main():          contacto1 = Contacto("Fulanoo", "99999999", "Calle A", "UNSA")          contacto2 = Contacto("Ciclano", "88888888", "Calle B", "TECSUP")          contacto3 = Contacto("Beltrano", "88889999", "Calle C", "Infancia")          agenda = Agenda()          agenda.add(contacto1)          agenda.add(contacto2)          agenda.add(contacto3)          print("Agenda inicial:")          print(agenda.\_\_str\_\_())          agenda.guardar()          contacto4 = Contacto("Fulano", "77777777", "Calle D", "UNSA")          agenda.add(contacto4)            print("\nAgenda después de añadir/modificar a Fulano:")          print(agenda.\_\_str\_\_())          agenda.remove("Ciclano")            print("\nAgenda después de eliminar a Ciclano:")          print(agenda.\_\_str\_\_())          agenda.guardar()          agenda.recuperar()          print("\nAgenda después de recuperar del archivo:")          print(agenda.\_\_str\_\_())  *if* \_\_name\_\_ == "\_\_main\_\_":      Principal.main()  *Agenda inicial:*  *Nombre: Fulanoo, Numero: 99999999, Dirección: Calle A, Relación: UNSA*  *Nombre: Ciclano, Numero: 88888888, Dirección: Calle B, Relación: TECSUP*  *Nombre: Beltrano, Numero: 88889999, Dirección: Calle C, Relación: Infancia*  *Contactos guardados correctamente.*  *Agenda después de añadir/modificar a Fulano:*  *Nombre: Fulanoo, Numero: 99999999, Dirección: Calle A, Relación: UNSA*  *Nombre: Ciclano, Numero: 88888888, Dirección: Calle B, Relación: TECSUP*  *Nombre: Beltrano, Numero: 88889999, Dirección: Calle C, Relación: Infancia*  *Nombre: Fulano, Numero: 77777777, Dirección: Calle D, Relación: UNSA*  *Contacto Ciclano eliminado.*  *Agenda después de eliminar a Ciclano:*  *Nombre: Fulanoo, Numero: 99999999, Dirección: Calle A, Relación: UNSA*  *Nombre: Beltrano, Numero: 88889999, Dirección: Calle C, Relación: Infancia*  *Nombre: Fulano, Numero: 77777777, Dirección: Calle D, Relación: UNSA*  *Contactos guardados correctamente.*  *Contactos recuperados correctamente.*  *Agenda después de recuperar del archivo:*  *Nombre: Fulanoo, Numero: 99999999, Dirección: Calle A, Relación: UNSA*  *Nombre: Beltrano, Numero: 88889999, Dirección: Calle C, Relación: Infancia*  *Nombre: Fulano, Numero: 77777777, Dirección: Calle D, Relación: UNSA*  ***Ejercicio 2:***  class ProductoStock:      def \_\_init\_\_(*self*, *nombre*, *valor*, *cantidad*):  *self*.nombre = *nombre*  *self*.valor = *valor*  *self*.cantidad = *cantidad*      def \_\_str\_\_(*self*):  *return* f"Producto: {*self*.nombre}, Valor: {*self*.valor}, Cantidad en stock: {*self*.cantidad}"  class StockProductos:      def \_\_init\_\_(*self*):  *self*.productos = {}      def adicionaProducto(*self*, *producto*):  *self*.productos[*producto*.nombre] = *producto*          print(f"Producto {*producto*.nombre} añadido al stock.")      def getProducto(*self*, *nombre*):  *return* *self*.productos.get(*nombre*, None)  class CarritoCompra:      def \_\_init\_\_(*self*, *stock*):  *self*.stock = *stock*  *self*.items = []      def adicionaItem(*self*, *nombre*, *cantidad*):          producto = *self*.stock.getProducto(*nombre*)  *if* producto and producto.cantidad >= *cantidad*:  *self*.items.append((producto, *cantidad*))              print(f"Item {*nombre*} añadido al carrito.")  *else*:              print(f"Stock insuficiente para {*nombre*} o producto no encontrado.")      def finalizarCompra(*self*):  *for* producto, cantidad *in* *self*.items:              producto.cantidad -= cantidad              print(f"Compra finalizada. Stock de {producto.nombre} actualizado a {producto.cantidad}.")      def calculaTotal(*self*):          total = sum(producto.valor \* cantidad *for* producto, cantidad *in* *self*.items)  *return* total  class Principal:      def main():            stock = StockProductos()          stock.adicionaProducto(ProductoStock("monitor", 500, 100))          stock.adicionaProducto(ProductoStock("teléfono", 150, 300))          stock.adicionaProducto(ProductoStock("teclado", 70, 50))          stock.adicionaProducto(ProductoStock("mouse", 50, 50))          carrito = CarritoCompra(stock)          carrito.adicionaItem("monitor", 59)          carrito.adicionaItem("teléfono", 99)          carrito.adicionaItem("teclado", 38)          carrito.finalizarCompra()            total = carrito.calculaTotal()          print(f"La suma de los productos: {total}")  *if* \_\_name\_\_ == "\_\_main\_\_":      Principal.main()  *Producto monitor añadido al stock.*  *Producto teléfono añadido al stock.*  *Producto teclado añadido al stock.*  *Producto mouse añadido al stock.*  *Item monitor añadido al carrito.*  *Item teléfono añadido al carrito.*  *Item teclado añadido al carrito.*  *Compra finalizada. Stock de monitor actualizado a 41.*  *Compra finalizada. Stock de teléfono actualizado a 201.*  *Compra finalizada. Stock de teclado actualizado a 12.*  *La suma de los productos: 47010* |
|

|  |
| --- |
| **REFERENCIAS Y BIBLIOGRAFÍA** |
| *---* |