Universidad Autónoma de Baja California
Lenguaje de Programación Python

Unidad 3: Métodos y Funciones

Nombre: Carlos Antonio Alvarez Angulo
Matrícula: 00366182
Grupo: Grupo 432
Fecha: 08/11/2023

MENU

1.- Agregar (automatico 10)

2.- Eliminar {ID}

3.- Imprimir lista (tabla)

4.- Buscar {ID}

5.- Buscar {appat} todas las coincidencias

6.- Ordenar {ID}

7.- Generar archivo {ID} (preguntar nombre del archivo)

a) excel

b) txt

c) cvs

d) MarkDown

8.- Cargar archivo {ID}

9.- Imprimir archivo {ID}

10.- Borrar Toda la lista {ID}

0.- SALIR

NOTA: Los datos del diccionario son los Datos básicos de un trabajador de una fabrica.

NOTA 2: VALIDAR EL PROGRAMA 100%

Preguntar siempre si esta seguro eliminar, No ordenar si ya esta Ordenado, decir si lista vacía, no existe o mostrar si estala búsqueda, No se permiten ID repetidos

```
[51] import pandas as pd
     import random
     from IPython.display import clear_output
```

```
[52] from google.colab import drive
     drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[72] ruta = "/content/drive/MyDrive/CAAA_PY_ACT12"
```

```
[54] import os
     path = "/content/drive/MyDrive/CAAA_PY_ACT12"
```

```
[55] import sys
     sys.path.append(ruta)
```

```
[56] trabajadores = {}
```

```
[57] def vali_num(numero):
         try:
             int(numero)
             return True
         except ValueError:
             return False
```

```
def registro():
    valores = []

    nombre_h = ["Juan", "Luis", "Carlos", "Roberto", "Pedro", "Abraham", "Tulio"]
    nombre_m = ["Sandra", "Lupita", "Maria", "Sonia", "Ana", "Vanessa", "Cristina"]
    apellidos = ["Villalobos", "Sanchez", "Perez", "Casas", "Paredes", "Alvarez", "Huerta", "Gutierrez", "Herrera", "Chavez", "Cota"]

    puesto = ["Supervisor", "Encargad@", "Conserje", "Gerente", "Guardia", "Empleado Gral", "Jefe"]
    sexo = ["HOMBRE", "MUJER"]

    valores.append(random.randint(10, 500))
    valores.append(random.choice(apellidos))
    valores.append(random.choice(apellidos))
    sex = random.randint(0, 1)
    if sex == 0:
        valores.append(random.choice(nombre_h))
    else:
        valores.append(random.choice(nombre_m))

    valores.append(random.choice(puesto))
    valores.append(sexo[sex])
    return valores
```

```python
[59] def agregar_trabajadores_automatico(cantidad=10):
        try:
            for _ in range(cantidad):
                datos = registro()
                id = datos[0]
                trabajadores[id] = {
                    'Nombre': datos[3],
                    'Apellido Paterno': datos[1],
                    'Apellido Materno': datos[2],
                    'Puesto': datos[4],
                    'Sexo': datos[5]
                }

            print(f"{cantidad} trabajadores agregados automáticamente.")
        except Exception as e:
            print(f"Error al agregar trabajadores automáticamente: {e}")
```

```python
[60] def eliminar_trabajador(id):
        if id in trabajadores:
            del trabajadores[id]
            print(f"Trabajador con ID {id} eliminado exitosamente.")
        else:
            print(f"No se encontró un trabajador con ID {id} en la lista.")
```

```python
def imprimir_lista():
    if not trabajadores:
        print("La lista de trabajadores está vacía.")
    else:
        print("| ID | Nombre | Apellido Paterno | Apellido Materno | Puesto | Sexo |")
        print("|----|--------|------------------|------------------|-------|------|")
        for id, datos in trabajadores.items():
            print(f"| {id} | {datos['Nombre']} | {datos['Apellido Paterno']} | {datos.get('Apellido Materno', '-')} | {datos.get('Puesto', '-')} | {datos.get('Sexo', '-')} |")
```

```python
[62] def buscar_por_id(id):
        if id in trabajadores:
            print(f"ID: {id}, Nombre: {trabajadores[id]['Nombre']}, Apellido Paterno: {trabajadores[id]['Apellido Paterno']}, "
                  f"Apellido Materno: {trabajadores[id]['Apellido Materno']}, Puesto: {trabajadores[id]['Puesto']}, "
                  f"Sexo: {trabajadores[id]['Sexo']}")
        else:
            print(f"No se encontró un trabajador con ID {id} en la lista.")
```

```python
[63] def buscar_por_appat(appat):
        coincidencias = [id for id, datos in trabajadores.items() if datos['Apellido Paterno'] == appat]
        if coincidencias:
            print("Coincidencias encontradas:")
            for id in coincidencias:
                datos = trabajadores[id]
                print(f"ID: {id}, Nombre: {datos['Nombre']}, Apellido Paterno: {datos['Apellido Paterno']}, "
                      f"Apellido Materno: {datos.get('Apellido Materno', '-')}, Puesto: {datos.get('Puesto', '-')}, "
                      f"Sexo: {datos.get('Sexo', '-')}")

        else:
            print(f"No se encontraron trabajadores con Apellido Paterno '{appat}'.")
```

```python
[64] def ordenar_lista():
        trabajadores_ordenados = dict(sorted(trabajadores.items(), key=lambda x: x[0]))
        trabajadores.clear()
        trabajadores.update(trabajadores_ordenados)
        print("Lista de trabajadores ordenada por ID.")
```

```
[65] def generar_archivo(extension):
         if not trabajadores:
             print("No se puede generar un archivo, la lista de trabajadores está vacía.")
             return

         nombre_archivo = input("Ingresa el nombre del archivo a generar: ")
         archivo = f"{ruta}/{nombre_archivo}"

         try:
             if extension == 'a':
                 archivo = f"{archivo}.xlsx"
                 df = pd.DataFrame.from_dict(trabajadores, orient='index')
                 df.to_excel(archivo, index_label='ID')
                 print(f"Archivo Excel '{archivo}' generado exitosamente.")
             elif extension == 'b':
                 archivo = f"{archivo}.txt"
                 with open(archivo, 'w') as f:
                     for id, datos in trabajadores.items():
                         f.write(f"ID: {id}, Nombre: {datos['Nombre']}, Apellido Paterno: {datos['Apellido Paterno']}, "
                                 f"Apellido Materno: {datos.get('Apellido Materno', '')}, Puesto: {datos.get('Puesto', '')}, "
                                 f"Sexo: {datos.get('Sexo', '')}\n")
                 print(f"Archivo de texto '{archivo}' generado exitosamente.")
             elif extension == 'c':
                 archivo = f"{archivo}.csv"
                 df = pd.DataFrame.from_dict(trabajadores, orient='index')
                 df.to_csv(archivo, index_label='ID')
                 print(f"Archivo CSV '{archivo}' generado exitosamente.")
             elif extension == 'd':
                 archivo = f"{archivo}.md"
                 with open(archivo, 'w') as f:
                     f.write("| ID | Nombre | Apellido Paterno | Apellido Materno | Puesto | Sexo |\n")
                     f.write("| --- | ------ | ---------------- | ---------------- | ------ | ---- |\n")
                     for id, datos in trabajadores.items():
                         f.write(f"| {id} | {datos['Nombre']} | {datos['Apellido Paterno']} | "
                                 f"{datos.get('Apellido Materno', '')} | {datos.get('Puesto', '')} | {datos.get('Sexo', '')} |\n")
                 print(f"Archivo Markdown '{archivo}' generado exitosamente.")
         except Exception as e:
             print(f"Error al generar el archivo: {e}")
```

```python
def cargar_archivo(extension):
    nombre_archivo = input("Ingresa el nombre del archivo a cargar: ")
    archivo = f"{ruta}/{nombre_archivo}"

    try:
        if extension == 'a':
            archivo = f"{archivo}.xlsx"
            df = pd.read_excel(archivo, index_col='ID')
            trabajadores.update(df.to_dict(orient='index'))
            print(f"Archivo Excel '{archivo}' cargado exitosamente.")
        elif extension == 'b':
            archivo = f"{archivo}.txt"
            with open(archivo, 'r') as f:
                for linea in f:
                    campos = linea.strip().split(', ')
                    id = int(campos[0].split(': ')[1])
                    datos_trabajador = {}
                    for campo in campos[1:]:
                        key, value = campo.split(': ')
                        datos_trabajador[key] = value
                    trabajadores[id] = datos_trabajador
                print(f"Archivo de texto '{archivo}' cargado exitosamente.")
        elif extension == 'c':
            archivo = f"{archivo}.csv"
            df = pd.read_csv(archivo, index_col='ID')
            trabajadores.update(df.to_dict(orient='index'))
            print(f"Archivo CSV '{archivo}' cargado exitosamente.")
        elif extension == 'd':
            archivo = f"{archivo}.md"
            with open(archivo, 'r') as f:
                lineas = f.readlines()
                header = [campo.strip() for campo in lineas[0].strip().split('|')[1:-1]]
                for linea in lineas[2:]:
                    campos = [campo.strip() for campo in linea.strip().split('|')[1:-1]]
                    datos_trabajador = dict(zip(header, campos))
                    id = int(datos_trabajador['ID'])
                    trabajadores[id] = datos_trabajador
                print(f"Archivo Markdown '{archivo}' cargado exitosamente.")
    except Exception as e:
        print(f"Error al cargar el archivo: {e}")
```

```
[67] def imprimir_archivo(extension):
        if not trabajadores:
            print("No se puede imprimir un archivo, la lista de trabajadores está vacía.")
            return

        nombre_archivo = input("Ingresa el nombre del archivo a imprimir: ")
        archivo = f"{ruta}/{nombre_archivo}"

        try:
            if extension == 'a':
                archivo = f"{archivo}.xlsx"
                df = pd.DataFrame.from_dict(trabajadores, orient='index')
                df.to_excel(archivo, index_label='ID')
                print(f"Archivo Excel '{archivo}' impreso exitosamente.")
            elif extension == 'b':
                archivo = f"{archivo}.txt"
                with open(archivo, 'w') as f:
                    for id, datos in trabajadores.items():
                        f.write(f"ID: {id}, Nombre: {datos['Nombre']}, Apellido Paterno: {datos['Apellido Paterno']}, "
                                f"Apellido Materno: {datos.get('Apellido Materno', '')}, Puesto: {datos.get('Puesto', '')}, "
                                f"Sexo: {datos.get('Sexo', '')}\n")
                print(f"Archivo de texto '{archivo}' impreso exitosamente.")
            elif extension == 'c':
                archivo = f"{archivo}.csv"
                df = pd.DataFrame.from_dict(trabajadores, orient='index')
                df.to_csv(archivo, index_label='ID')
                print(f"Archivo CSV '{archivo}' impreso exitosamente.")
            elif extension == 'd':
                archivo = f"{archivo}.md"
                with open(archivo, 'w') as f:
                    f.write("| ID | Nombre | Apellido Paterno | Apellido Materno | Puesto | Sexo |\n")
                    f.write("| --- | ------ | ---------------- | ---------------- | ------ | ---- |\n")
                    for id, datos in trabajadores.items():
                        f.write(f"| {id} | {datos['Nombre']} | {datos['Apellido Paterno']} | "
                                f"{datos.get('Apellido Materno', '')} | {datos.get('Puesto', '')} | {datos.get('Sexo', '')} |\n")
                print(f"Archivo Markdown '{archivo}' impreso exitosamente.")
        except Exception as e:
            print(f"Error al imprimir el archivo: {e}")
```

```
[68] def borrar_toda_la_lista():
        trabajadores.clear()
        print("Lista de trabajadores borrada exitosamente.")
```

```python
def main():
    while True:
        clear_output()
        print("\nMENU")
        print("1.- Agregar (automático 10)")
        print("2.- Eliminar {ID}")
        print("3.- Imprimir lista (tabla)")
        print("4.- Buscar {ID}")
        print("5.- Buscar {appat} todas las coincidencias")
        print("6.- Ordenar {ID}")
        print("7.- Generar archivo {ID} (preguntar nombre del archivo)")
        print("   a) Excel")
        print("   b) Texto")
        print("   c) CSV")
        print("   d) Markdown")
        print("8.- Cargar archivo {ID}")
        print("9.- Imprimir archivo {ID}")
        print("10.- Borrar toda la lista {ID}")
        print("0.- SALIR")

        opcion = input("Selecciona una opción: ")
```

```
if opcion == '1':
    clear_output()
    agregar_trabajadores_automatico()
    input("Presione una tecla para continuar...")
elif opcion == '2':
    clear_output()
    id = input("Ingresa el ID del trabajador a eliminar: ")
    if vali_num(id):
        id = int(id)
        eliminar_trabajador(id)
        input("Presione una tecla para continuar...")
    else:
        print("ID no válido. Por favor, ingresa un número entero.")
        input("Presione una tecla para continuar...")
elif opcion == '3':
    clear_output()
    imprimir_lista()
    input("Presione una tecla para continuar...")
elif opcion == '4':
    clear_output()
    id = input("Ingresa el ID del trabajador a buscar: ")
    if vali_num(id):
        id = int(id)
        buscar_por_id(id)
        input("Presione una tecla para continuar...")
    else:
        print("ID no válido. Por favor, ingresa un número entero.")
        input("Presione una tecla para continuar...")
elif opcion == '5':
    clear_output()
    appat = input("Ingresa el Apellido Paterno a buscar: ")
    buscar_por_appat(appat)
    input("Presione una tecla para continuar...")
elif opcion == '6':
    clear_output()
    ordenar_lista()
    input("Presione una tecla para continuar...")
elif opcion == '7':
    clear_output()
    extension = input("Selecciona la extensión del archivo (a, b, c, d): ").lower()
    if extension in ['a', 'b', 'c', 'd']:
        generar_archivo(extension)
```

```
                generar_archivo(extension)
            else:
                print("Extensión no válida. Por favor, selecciona 'a', 'b', 'c' o 'd'.")
            input("Presione una tecla para continuar...")
        elif opcion == '8':
            clear_output()
            extension = input("Selecciona la extensión del archivo a cargar (a, b, c, d): ").lower()
            if extension in ['a', 'b', 'c', 'd']:
                cargar_archivo(extension)
            else:
                print("Extensión no válida. Por favor, selecciona 'a', 'b', 'c' o 'd'.")
            input("Presione una tecla para continuar...")
        elif opcion == '9':
            clear_output()
            extension = input("Selecciona la extensión del archivo a imprimir (a, b, c, d): ").lower()
            if extension in ['a', 'b', 'c', 'd']:
                imprimir_archivo(extension)
            else:
                print("Extensión no válida. Por favor, selecciona 'a', 'b', 'c' o 'd'.")
            input("Presione una tecla para continuar...")
        elif opcion == '10':
            clear_output()
            borrar_toda_la_lista()
            input("Presione una tecla para continuar...")
        elif opcion == '0':
            clear_output()
            break
        else:
            print("Opción no válida. Por favor, selecciona una opción del menú.")
```

```
if __name__ == '__main__':
    main()
```

```
MENU
1.- Agregar (automático 10)
2.- Eliminar {ID}
3.- Imprimir lista (tabla)
4.- Buscar {ID}
5.- Buscar {appat} todas las coincidencias
6.- Ordenar {ID}
7.- Generar archivo {ID} (preguntar nombre del archivo)
   a) Excel
   b) Texto
   c) CSV
   d) Markdown
8.- Cargar archivo {ID}
9.- Imprimir archivo {ID}
10.- Borrar toda la lista {ID}
0.- SALIR
Selecciona una opción:
```

```
| ID | Nombre | Apellido Paterno | Apellido Materno | Puesto | Sexo |
|----|--------|------------------|------------------|--------|------|
| 500 | Luis | Cota | Casas | Conserje | HOMBRE |
| 459 | Juan | Sanchez | Perez | Conserje | HOMBRE |
| 262 | Luis | Cota | Cota | Empleado Gral | HOMBRE |
| 209 | Cristina | Sanchez | Villalobos | Gerente | MUJER |
| 116 | Abraham | Paredes | Gutierrez | Encargad@ | HOMBRE |
| 448 | Lupita | Herrera | Villalobos | Jefe | MUJER |
| 78 | Abraham | Cota | Casas | Gerente | HOMBRE |
| 359 | Roberto | Casas | Villalobos | Guardia | HOMBRE |
| 491 | Lupita | Villalobos | Gutierrez | Empleado Gral | MUJER |
| 469 | Sonia | Villalobos | Paredes | Encargad@ | MUJER |
Presione una tecla para continuar...
```