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: 29/10/2023

REALIZA UN PROGRAMA QUE UTILICE UNA LISTA DE DICCIONARIOS

MENU

1.- Agregar (automatico)

2.- Agregar (manual)

3.- Imprimir lista

a)

b)

c)

4.- Buscar {ID}

5.- Ordenar

6.- Eliminar {ID}

7.- 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

```
[2] !pip install faker
```

```
Collecting faker
  Downloading Faker-19.12.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 18.3 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.10/dist-packages (from faker) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.4->faker) (1.16.0)
Installing collected packages: faker
Successfully installed faker-19.12.0
```

```
[4] !pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
[5] import random
    import json
    import yaml
    import pandas as pd
    from faker import Faker
    from IPython.display import clear_output
```

```
[6] # Función para generar IDs únicos
    def generar_id_unico(trabajadores):
        while True:
            nuevo_id = str(random.randint(300000, 399999))
            if nuevo_id not in [trabajador["ID"] for trabajador in trabajadores]:
                return nuevo_id
```

```python
# Función para agregar un trabajador automáticamente
def agregar_trabajador_automatico(trabajadores, faker):
    clear_output()
    nuevo_trabajador = {
        "ID": generar_id_unico(trabajadores),
        "Nombre": faker.first_name(),
        "Apellido": faker.last_name(),
        "Edad": random.randint(25, 60),
        "Cargo": faker.job()
    }
    trabajadores.append(nuevo_trabajador)
    print("Trabajador agregado automáticamente con ID:", nuevo_trabajador["ID"])
```

```python
# Función para agregar un trabajador manualmente
def agregar_trabajador_manual(trabajadores):
    clear_output()
    trabajadores_iniciales = [
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "Juan",
            "Apellido": "García",
            "Edad": 30,
            "Cargo": "Operario"
        },
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "María",
            "Apellido": "López",
            "Edad": 28,
            "Cargo": "Técnico"
        },
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "Luis",
            "Apellido": "Martínez",
            "Edad": 35,
            "Cargo": "Supervisor"
        },
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "Ana",
            "Apellido": "Rodríguez",
            "Edad": 32,
            "Cargo": "Operario"
        },
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "Carlos",
            "Apellido": "Sánchez",
            "Edad": 27,
            "Cargo": "Técnico"
        },
```

```
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Laura",
        "Apellido": "Gómez",
        "Edad": 31,
        "Cargo": "Operario"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Javier",
        "Apellido": "Hernández",
        "Edad": 29,
        "Cargo": "Técnico"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Sofía",
        "Apellido": "Díaz",
        "Edad": 34,
        "Cargo": "Supervisor"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Ricardo",
        "Apellido": "Pérez",
        "Edad": 33,
        "Cargo": "Operario"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Valentina",
        "Apellido": "Torres",
        "Edad": 26,
        "Cargo": "Técnico"
    },
```

```
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Alejandro",
        "Apellido": "Mendoza",
        "Edad": 29,
        "Cargo": "Operario"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Carmen",
        "Apellido": "Vargas",
        "Edad": 36,
        "Cargo": "Supervisor"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Roberto",
        "Apellido": "Castillo",
        "Edad": 30,
        "Cargo": "Operario"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Elena",
        "Apellido": "Fernández",
        "Edad": 29,
        "Cargo": "Técnico"
    },
    {
        "ID": generar_id_unico(trabajadores),
        "Nombre": "Miguel",
        "Apellido": "Rojas",
        "Edad": 31,
        "Cargo": "Operario"
    },
```

```
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "Jorge",
            "Apellido": "Lara",
            "Edad": 32,
            "Cargo": "Operario"
        },
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "Carmen",
            "Apellido": "Soto",
            "Edad": 31,
            "Cargo": "Técnico"
        },
        {
            "ID": generar_id_unico(trabajadores),
            "Nombre": "Alejandro",
            "Apellido": "Dominguez",
            "Edad": 28,
            "Cargo": "Operario"
        }
    ]

    trabajadores.extend(trabajadores_iniciales)
    print("Trabajadores agregados manualmente.")
    input("Presione una tecla para continuar...")
```

```
# Función para imprimir la lista de trabajadores en formato PANDAS
def imprimir_lista_pandas(trabajadores):
    clear_output()
    if not trabajadores:
        print("La lista está vacía.")
    else:
        print("Lista de trabajadores en formato de tabla:")
        df = pd.DataFrame(trabajadores)
        print(df)
    input("Presione una tecla para continuar...")

# Función para imprimir la lista de trabajadores en formato JSON
def imprimir_lista_json(trabajadores):
    clear_output()
    if not trabajadores:
        print("La lista está vacía.")
    else:
        print("Lista de trabajadores en formato JSON:")
        print(json.dumps(trabajadores, indent=4))
    input("Presione una tecla para continuar...")

# Función para imprimir la lista de trabajadores en formato YAML
def imprimir_lista_yaml(trabajadores):
    clear_output()
    if not trabajadores:
        print("La lista está vacía.")
    else:
        print("Lista de trabajadores en formato YAML:")
        print(yaml.dump(trabajadores))
    input("Presione una tecla para continuar...")
```

```python
# Función para buscar un trabajador por su ID
def buscar_trabajador_id(trabajadores, id_buscar):
    clear_output()
    for trabajador in trabajadores:
        if trabajador["ID"] == id_buscar:
            return trabajador
    input("Presione una tecla para continuar...")
    return None
```

```python
[22] # Función para ordenar la lista de trabajadores por ID
def ordenar_lista(trabajadores):
    clear_output()
    trabajadores.sort(key=lambda x: x["ID"])
    input("Presione una tecla para continuar...")
```

```python
[23] # Función para eliminar un trabajador por su ID
def eliminar_trabajador_id(trabajadores, id_eliminar):
    clear_output()
    trabajador = buscar_trabajador_id(trabajadores, id_eliminar)
    if trabajador:
        trabajadores.remove(trabajador)
        print("Trabajador con ID", id_eliminar, "eliminado.")
    else:
        print("No se encontró un trabajador con ID", id_eliminar)
    input("Presione una tecla para continuar...")
```

```python
[24] # Función para borrar toda la lista de trabajadores
def borrar_lista(trabajadores):
    clear_output()
    trabajadores.clear()
    print("Lista de trabajadores borrada.")
    input("Presione una tecla para continuar...")
```

```python
# Generar la base de datos de los 50 trabajadores
trabajadores = []
faker = Faker()
for _ in range(50):
    agregar_trabajador_automatico(trabajadores, faker)
clear_output()
```

```python
# Menú principal
def menu():
    while True:
        print("\nMENU\n")
        print("1.- Agregar (automático)")
        print("2.- Agregar (manual)")
        print("3.- Imprimir lista (a)")
        print("4.- Imprimir lista (b)")
        print("5.- Imprimir lista (c)")
        print("6.- Buscar {ID}")
        print("7.- Ordenar")
        print("8.- Eliminar {ID}")
        print("9.- Borrar Toda la lista")
        print("0.- SALIR")

        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            agregar_trabajador_automatico(trabajadores, faker)
            input("Presione una tecla para continuar...")
        elif opcion == "2":
            agregar_trabajador_manual(trabajadores)
        elif opcion == "3":
            imprimir_lista_pandas(trabajadores)
        elif opcion == "4":
            imprimir_lista_yaml(trabajadores)
        elif opcion == "5":
            imprimir_lista_json(trabajadores)
        elif opcion == "6":
            id_buscar = input("Ingrese el ID a buscar: ")
            trabajador = buscar_trabajador_id(trabajadores, id_buscar)
            if trabajador:
                print("Trabajador encontrado:")
                print(json.dumps(trabajador, indent=4))
            else:
                print("No se encontró un trabajador con ID", id_buscar)
        elif opcion == "7":
            ordenar_lista(trabajadores)
            print("Lista de trabajadores ordenada por ID.")
        elif opcion == "8":
            id_eliminar = input("Ingrese el ID del trabajador a eliminar: ")
            eliminar_trabajador_id(trabajadores, id_eliminar)
        elif opcion == "9":
            confirmacion = input("¿Está seguro de que desea borrar toda la lista? (S/N): ")
            if confirmacion.lower() == "s":
                borrar_lista(trabajadores)
            else:
                print("Operación cancelada.")
        elif opcion == "0":
            print("Saliendo del programa.")
            break
        else:
            print("Opción no válida. Intente nuevamente.")
```

```
menu()
```

```
Lista de trabajadores borrada.
Presione una tecla para continuar...

MENU

1.- Agregar (automático)
2.- Agregar (manual)
3.- Imprimir lista (a)
4.- Imprimir lista (b)
5.- Imprimir lista (c)
6.- Buscar {ID}
7.- Ordenar
8.- Eliminar {ID}
9.- Borrar Toda la lista
0.- SALIR
Seleccione una opción: 0
Saliendo del programa.
```