Escuela Politécnica Nacional | EDA II-Informe N°2

Ismael Freire

Tabla de contenidos

1. Objetivos

- Implementar funciones, listas, diccionarios, y estructuras de control de decisión y repetición para resolver problemas de manera eficiente y estructurada.
- Desarrollar un programa en Python que permita gestionar una lista de tareas de una fiesta, indicando qué actividades han sido completadas y ocultándolas de la lista.
- Crear un programa que analice los gastos mensuales de papelería de una empresa, identificando el mayor y menor gasto, así como calculando el promedio anual.

2. Introducción

En el presente informe se tratarán dos problemas prácticos, el primero de ellos consiste en gestionar una lista de actividades para la organización de un evento, donde el usuario es capaz de añadir, marcar la actividad como check o pendiente y actualizar la lista en consecuencia. El segundo se enfoca en el gasto mensual de una papelería donde se debe identificar los meses con mayor y menor gasto, asi como calcular el promedio anual.

Ambos programas implementan estructuras de control como listas, diccionarios y ciclos, las cuales permiten dirigir el flujo de ejecución del programa de manera adecuada. Además, tanto

las listas como diccionarios que son estructuras clave para el almacenamiento y gestión de elementos permiten un correcto desarrollo de los problemas que se han planteado.

3. Ejercicios planteados y/o programas implementados

- Ejercicio 1: Tienes una fiesta y haces una lista de tipo "TO-DOs" para preparar todas las actividades, como por ejemplo: "bebidas", "desechables", "parlantes", "globos", etc. Has un programa que barra la lista de actividad en actividad y tú puedas indicar si está listo o no (check o no). Si hay una actividad con "check", que ya no se muestre la próxima vez que revisas tu lista de TO-DOs

Para el desarrollo de esta pregunta se usaron tres funciones que permitieron manejar una lista de actividades.

Primero se tiene a *llenar_dict ()*, que es la función encargada de crear un diccionario en la que cada clave es una actividad y su valor asociado es "pendiente". Se inicia con un diccionario vacío *dic_full*, luego con un bucle *for* el usuario ingresa tantas actividades como ha especificado *num_activities* . Seguidamente, a cada actividad ingresada se le asigna un valor por defecto "pendiente" y después de llenar el diccionario, se imprimen las actividades ingresadas y se lo retorna con las actividades y su estado.

```
def llenar_dict (num_activities):
    dic_full = dict()
    for i in range (num_activities):
        key = input(f"Ingrese una actividad {i+1}: ")
        value = "pendiente"
        dic_full[key] = value
    lista_keys = list(dic_full.keys())
    print("Su lista actual de actividades pendientes es->")
    print(lista_keys)
    return dic_full
```

Para la función *check_dict ()* se toma como parámetro un diccionario *dictionary*, que contiene a las actividades, claves, y a su estado actual como valor. Después, se recorre cada actividad y se pregunta al usuario si ha sido realizada "check" o si sigue pendiente, en el caso de ingresar "check" se elimina la clave del diccionario para evitar mostrarla la próxima vez, caso contrario, se sigue manteniendo la clave. Al final, se retorna el diccionario actualizado.

```
def check_dict (dictionary):
    new_dict = dict()
    print("Verifique que tarea se ha realizado")
    for i in range(len(dictionary)) :
```

```
        lista_keys = list(dictionary.keys())
        print("De la tarea {}".format(lista_keys[i].upper()), end = " ")
        new_dict[lista_keys[i]] = str(input("ingrese check o pendiente->").lower())
        if new_dict[lista_keys[i]] == "check":
            del new_dict[lista_keys[i]]
        else:
            pass
    return new_dict
```

La función *lista ()* corresponde a la función principal, que gestiona la lista de actividades mediante un menú de acciones del que el usuario puede elegir.

```
def lista ():
    bandera = True
    dic = dict()
    print("¡Bienvenido al sistema!")
    while bandera != False :
        print("\n1. Llenar la lista")
        print("2. Marcar valores de la lista")
        print("3. Revisar lista")
        print("4. Salir")
        opcion = int(input("\nElija una opcion: "))
        if opcion == 1:
            dic = llenar_dict(int(input("Ingrese el tamanio de la nueva lista->")))
        elif opcion == 2:
            if not dic :
                print("No existen valores en la lista")
            else:
                dic = check_dict(dic)
        elif opcion == 3:
            if not dic:
                print("No existen cambios o valores en la lista")
            else:
                print("Su lista actual es->")
                print(dic)
        elif opcion == 4:
            bandera = False
```

Finalmente, se llama la función principal *lista ()* y se ejecuta todo el código respectivo.

```
lista()
```

```
¡Bienvenido al sistema!

1. Llenar la lista
2. Marcar valores de la lista
3. Revisar lista
4. Salir


Elija una opcion:  3

No existen cambios o valores en la lista

1. Llenar la lista
2. Marcar valores de la lista
3. Revisar lista
4. Salir


Elija una opcion:  2

No existen valores en la lista

1. Llenar la lista
2. Marcar valores de la lista
3. Revisar lista
4. Salir


Elija una opcion:  1
Ingrese el tamanio de la nueva lista-> 3
Ingrese una actividad 1:  bebidas
Ingrese una actividad 2:  desechables
Ingrese una actividad 3:  parlantes

Su lista actual de actividades pendientes es->
['bebidas', 'desechables', 'parlantes']

1. Llenar la lista
```

```
2. Marcar valores de la lista
3. Revisar lista
4. Salir


Elija una opcion:  3

Su lista actual es->
{'bebidas': 'pendiente', 'desechables': 'pendiente', 'parlantes': 'pendiente'}

1. Llenar la lista
2. Marcar valores de la lista
3. Revisar lista
4. Salir


Elija una opcion:  2

Verifique que tarea se ha realizado
De la tarea BEBIDAS

ingrese check o pendiente-> check

De la tarea DESECHABLES

ingrese check o pendiente-> pendiente

De la tarea PARLANTES

ingrese check o pendiente-> check


1. Llenar la lista
2. Marcar valores de la lista
3. Revisar lista
4. Salir


Elija una opcion:  3
```

```
Su lista actual es->
{'desechables': 'pendiente'}

1. Llenar la lista
2. Marcar valores de la lista
3. Revisar lista
4. Salir


Elija una opcion:  4
```

- Ejercicio 2: Cada mes tu empresa gasta en papelería cierto monto. Haz un programa que permita verificar en qué mes se gastó más y el valor, en qué mes se gastó menos y el valor, y finalmente el promedio de gasto en el año.

En esta pregunta, se presenta la función *mes*, que contiene los nombres de los meses del año. Dentro, genera un diccionario vacío *gasto_mes* donde cada clave es un mes y su valor es el gasto correspondiente ingresado por el usuario. A través, de un bucle *for*, se recorre la lista de meses y, por cada uno, se solicita al usuario que ingrese el gasto asociado, el cual se lo guarda como un valor flotante. Al final, se imprime el diccionario completo y se retorna para su uso posterior.

```python
def mes ():
    list_mes = ["Enero", "Febrero", "Marzo",
                "Abril", "Mayo", "Junio",
                "Julio", "Agosto", "Septiembre",
                "Octubre", "Noviembre", "Diciembre"]
    gasto_mes = dict()
    print("Ingrese los gastos por mes:")
    for i in range(len(list_mes)):
        #print(list_mes[i])
        print(f"{list_mes[i]}: ", end="")
        gasto_mes[list_mes[i]] = float(input())
    print("\nLos gastos de cada mes fueron->")
    lista_dict = list(gasto_mes.items())
    primera_mitad = dict(lista_dict[:4])
    segunda_mitad = dict(lista_dict[4:4*2])
    tercera_mitad = dict(lista_dict[4*2:])
    print(primera_mitad)
    print(segunda_mitad)
    print(tercera_mitad)
    return gasto_mes
```

La siguiente función *mayor_promedio()*, que toma como parámetros el diccionario gasto_mes. Primero, obtiene los valores "gastos" del diccionario y calcula el mayor y el menor, usando las funciones *max ()* y *min()*. Además, calcula el promedio anual de los gastos redondeandolo a dos decimales. Luego, recorre el diccionario y encuentra los meses correspondientes al mayor y menor gasto. Finalmente, la función imprime un resumen que indica en qué mes se produjo el mayor y el menor gasto, junto con los montos y el promedio anual de gastos.

```
def mayor_promedio (gasto_mes):
    mes_max = ""
    mes_min = ""
    list_gasto = list(gasto_mes.values())
    gasto_max = max(list_gasto)
    gasto_min = min(list_gasto)
    gasto_promedio = round((sum(list_gasto)/len(list_gasto)), 2)
    for key, value in gasto_mes.items():
        if value == gasto_max:
            mes_max = key
        elif value == gasto_min:
            mes_min = key
        else:
            pass
    print("\nEl mayor gasto fue en el mes de {} y es ${}.".format(mes_max, gasto_max))
    print("El menor fue en el mes de {} y es ${}.".format(mes_min, gasto_min))
    print("El promedio gastado en todo el año es ${}".format(gasto_promedio))
```

A continuación, se presenta la ejecución del programa:

```
valores = mes()
mayor_promedio(valores)
```

```
Ingrese los gastos por mes:
Enero:

 120

Febrero:

 125

Marzo:
```

136

Abril:

178

Mayo:

98

Junio:

654

Julio:

145

Agosto:

169

Septiembre:

654

Octubre:

324

Noviembre:

698

Diciembre:

1000

```
Los gastos de cada mes fueron->
{'Enero': 120.0, 'Febrero': 125.0, 'Marzo': 136.0, 'Abril': 178.0}
{'Mayo': 98.0, 'Junio': 654.0, 'Julio': 145.0, 'Agosto': 169.0}
{'Septiembre': 654.0, 'Octubre': 324.0, 'Noviembre': 698.0, 'Diciembre': 1000.0}

El mayor gasto fue en el mes de Diciembre y es $1000.0.
El menor fue en el mes de Mayo y es $98.0.
El promedio gastado en todo el año es $358.42
```

4. Conclusiones

Los ejercicios resueltos destacan la importancia de las estructuras de control y las funciones en la organización y eficiencia del código. El uso de listas y diccionarios simplificó la gestión de datos, como la planificación de actividades para una fiesta y el análisis de gastos mensuales. Las declaraciones condicionales y los bucles automatizaron las decisiones, haciendo el programa más dinámico y eficaz. La definición de funciones resultó fundamental para estructurar el código en bloques independientes y específicos, lo que facilitó la comprensión y cualquier cambio a realizarse al asignar a cada función una tarea particular dentro del programa.

5. Referencias bibliográficas

B. S. Marco, «Introducción a la programación con Python,» 28 03 2023. [En línea]. Available: https://www.mclibre.org/consultar/python/lecciones/python-operaciones-matematicas.html. [Último acceso: 12 10 2024].

6. Declaración uso de IA

En el presente informe la IA fue implementada para aplicar nuevas funciones como lo son min() y max() que obtienen el mínimo y máximo valor de una lista. También para la aplicación de la función round() que permitió obtener un promedio redondeado de hasta dos decimales, y para la división .