


Trabajo práctico nro. 6

	Asignatura: Programación I	
	Cursado: Primer Trimestre	Horas semanales: 4
	Carrera: <i>Tecnicatura Universitaria en Programación</i>	Nivel (Año): <input type="checkbox"/> 1° <input type="checkbox"/> 2° <input type="checkbox"/> 3°
	Ciclo Lectivo: 2023	

Integrantes de la Cátedra:

- DOCENTES:

Nombre del Profesor	Periodo	Cantidad horas materia
		6 horas

1. Solicitar al usuario que ingrese números, estos deben guardarse en una lista. Para finalizar el usuario debe ingresar 0, el cual no debe guardarse.

```
1 # Inicializar una lista para almacenar los números ingresados
2 numbers = []
3
4 while True:
5     number = int(input("Ingresa un número (ingresa 0 para finalizar): "))
6
7     if number == 0:
8         break # Salir del bucle al ingresar 0
9     else:
10        numbers.append(number)
11
12 print("Números ingresados (sin incluir el 0):", numbers)
13
```

2. A continuación, solicitar al usuario que ingrese un número y, si el número está en la lista, eliminar su primera ocurrencia. Mostrar un mensaje si no es posible eliminar.

```
# Inicializar una lista para almacenar los números ingresados
numbers = []
while True:
    number = int(input("Ingresa un número (ingresa 0 para finalizar): "))

    if number == 0:
        break # Salir del bucle al ingresar 0
    else:
        numbers.append(number)

print("Números ingresados (sin incluir el 0):", numbers)

# Solicitar al usuario un número a eliminar
number_to_delete = int(input("Ingresa un número a eliminar: "))

if number_to_delete in numbers:
    numbers.remove(number_to_delete)
    print(f"Se eliminó la primera ocurrencia de {number_to_delete} en la lista.")
    print("Lista actualizada:", numbers)
else:
    print(f"{number_to_delete} no se encuentra en la lista. No es posible eliminarlo.")
```

3. Imprimir la sumatoria de todos los números de la lista.

```
1 # Inicializar una lista para almacenar los números ingresados
2 numbers = []
3
4 while True:
5     number = int(input("Ingresa un número (ingresa 0 para finalizar): "))
6
7     if number == 0:
8         break # Salir del bucle al ingresar 0
9     else:
10        numbers.append(number)
11
12 print("Números ingresados (sin incluir el 0):", numbers)
13
14 # Sumar todos los números en la lista
15 total_sum = sum(numbers)
16
17 print(f"La sumatoria de todos los números en la lista es: {total_sum}")
```

4. Solicitar al usuario otro número y crear una lista con los elementos de la lista original, que sean menores que el número dado. Imprimir esta nueva lista, iterando por ella.

```
# Inicializar una lista para almacenar los números ingresados
numbers = []
while True:
    number = int(input("Ingresa un número (ingresa 0 para finalizar): "))
    if number == 0:
        break # Salir del bucle al ingresar 0
    else:
        numbers.append(number)
print("Números ingresados (sin incluir el 0):", numbers)
# Solicitar al usuario un número para filtrar la lista
filter_num = int(input("Ingresa un número para filtrar la lista: "))
# Crear una nueva lista con los elementos menores que el número dado
new_list = [num for num in numbers if num < filter_num]
print("Nueva lista con elementos menores que", filter_num, ":", new_list)
# Imprimir la nueva lista iterando por ella
print("Elementos de la nueva lista:")
for elemento in new_list:
    print(elemento)
```

5. Generar e imprimir una nueva lista que contenga como elementos a tuplas, cada una compuesta por un número de la lista original y la cantidad de veces que aparece en ella. Por ejemplo, si la lista original es [5,16,2,5,57,5,2], la nueva lista contendrá: [(5,3),(16,1),(2,2),(57,1)]

```
# Inicializar una lista para almacenar los números ingresados
numbers = []
while True:
    number = int(input("Ingresa un número (ingresa 0 para finalizar): "))
    if number == 0:
        break # Salir del bucle al ingresar 0
    else:
        numbers.append(number)
print("Números ingresados (sin incluir el 0):", numbers)
# Crear un diccionario para contar las ocurrencias de cada número
coincidences = {}
for number in numbers:
    if number in coincidences:
        coincidences[number] += 1
    else:
        coincidences[number] = 1
# Crear una nueva lista de tuplas con los números y sus ocurrencias
new_list = [(number, coincidences) for number, coincidences in coincidences.items()]
print("Nueva lista de tuplas con ocurrencias:")
print(new_list)
```

6. Solicitar al usuario que ingrese los nombres de pila de los alumnos de nivel primario de una escuela, finalizando al ingresar 'x'. A continuación, solicitar que ingrese los nombres de los alumnos de nivel secundario, finalizando al ingresar 'x'.
- Informar los nombres de todos los alumnos de nivel primario y de los de nivel secundario, sin repeticiones.
 - Informar qué nombres se repiten entre los alumnos de nivel primario y secundario.
 - Informar qué nombres de nivel primario no se repiten en los de nivel secundario.

```

# Inicializar listas para almacenar los nombres de nivel primario y secundario
primary_names = []
secondary_names = []
# Solicitar los nombres de nivel primario
print("Ingresa los nombres de los alumnos de nivel primario (ingresa 'x' para finalizar):")
while True:
    name = input()
    if name == 'x':
        break
    primary_names.append(name)
# Solicitar los nombres de nivel secundario
print("Ingresa los nombres de los alumnos de nivel secundario (ingresa 'x' para finalizar):")
while True:
    name = input()
    if name == 'x':
        break
    secondary_names.append(name)
# Informar los nombres sin repeticiones
primary_names = list(set(primary_names))
secondary_names = list(set(secondary_names))
print("\nNombres de alumnos de nivel primario:")
for name in primary_names:
    print(name)
print("\nNombres de alumnos de nivel secundario:")
for name in secondary_names:
    print(name)
# Informar los nombres que se repiten
repeated = list(set(primary_names) & set(secondary_names))
print("\nNombres que se repiten entre primario y secundario:")
for name in repeated:
    print(name)
# Informar los nombres de nivel primario que no se repiten en secundario
no_repeated_primary = [name for name in primary_names if name not in secondary_names]
print("\nNombres de nivel primario que no se repiten en secundario:")
for name in no_repeated_primary:
    print(name)

```

7. Escribir un programa que procese strings ingresados por el usuario. La lectura finaliza cuando se hayan procesado 50 strings. Al finalizar, informar la cantidad total de ocurrencias de cada carácter, en todos los strings ingresados. Ejemplo:

```

'r':5
'%':3
'a':8
'9':1

```

```

# Inicializar un diccionario para contar las ocurrencias de cada carácter
coincidences = {}
# Contador para llevar el registro de la cantidad de strings procesados
count_strings = 0
while count_strings < 50:
    # Solicitar un string al usuario
    into_str = input("Ingresa un string (o 'x' para finalizar): ")
    if into_str == 'x':
        break # Salir del bucle al ingresar 'x'
    # Actualizar el contador de strings
    count_strings += 1
    # Contar las ocurrencias de cada carácter en el string
    for character in into_str:
        if character in coincidences:
            coincidences[character] += 1
        else:
            coincidences[character] = 1
    # Imprimir la cantidad total de ocurrencias de cada carácter
    print("Cantidad total de ocurrencias de cada carácter:")
    for character, quantity in coincidences.items():
        print(f"'{character}': {quantity}")

```

8. Diez equipos de la liga inter-barrial identificados con los números 1, 2, 3, 4, ..., 10, participaron en un campeonato de fútbol con modalidad todos contra todos.

Los goles anotados en cada encuentro se registraron en el siguiente cuadro:

Goles(F,C)	1	2	3	4	...	10
1	0	4	2	1	...	
2	5	0	3	2	...	
3	0	2	0	1	...	
4	1	0	2	0	...	
...	
10						0

Escriba un programa que:

- Lea el cuadro de goles en un arreglo de dos dimensiones.
- Muestre para cada equipo la cantidad de triunfos, empates y derrotas.
- Muestre la diferencia entre el total de goles marcados y el total de goles recibidos.
- Determine la cantidad de puntos obtenidos por cada equipo.

```

sers > Tity Lacoste > OneDrive > Escritorio > x=46.py > ...
# Inicializar el cuadro de goles
goles = [
    [0, 4, 2, 1, 3, 0, 1, 2, 1, 4],
    [3, 0, 2, 3, 1, 2, 2, 2, 0, 1],
    [1, 2, 0, 1, 0, 3, 1, 0, 4, 0],
    [4, 2, 3, 0, 1, 2, 0, 3, 2, 0],
    [2, 3, 0, 1, 0, 1, 4, 0, 2, 3],
    [0, 1, 2, 2, 1, 0, 2, 1, 3, 0],
    [2, 2, 3, 0, 4, 1, 0, 1, 1, 3],
    [1, 1, 0, 2, 0, 3, 0, 0, 1, 2],
    [2, 2, 3, 0, 3, 2, 1, 0, 0, 2],
    [0, 3, 1, 0, 3, 1, 3, 2, 2, 0]
]

# Inicializar listas para llevar el registro de triunfos, empates, derrotas, goles marcados y goles recibidos
triunfos = [0] * 10
empates = [0] * 10
derrotas = [0] * 10
goles_marcados = [0] * 10
goles_recibidos = [0] * 10

# Calcular los resultados y registrar los datos
for i in range(10):
    for j in range(10):
        if i != j:
            if goles[i][j] > goles[j][i]:
                triunfos[i] += 1
            elif goles[i][j] < goles[j][i]:
                derrotas[i] += 1
            else:
                empates[i] += 1
            goles_marcados[i] += goles[i][j]
            goles_recibidos[i] += goles[j][i]

# Calcular los puntos obtenidos por cada equipo (3 puntos por triunfo, 1 punto por empate)
puntos = [triunfos[i] * 3 + empates[i] for i in range(10)]

# Imprimir los resultados
for equipo in range(10):
    print(f"Equipo {equipo + 1}:")
    print(f"Triunfos: {triunfos[equipo]}, Empates: {empates[equipo]}, Derrotas: {derrotas[equipo]}")
    print(f"Diferencia de goles: {goles_marcados[equipo] - goles_recibidos[equipo]}")
    print(f"Puntos obtenidos: {puntos[equipo]}")
    print()

```

9. Escribir un programa que simule el juego clásico de Memoria (Memotest) utilizando matrices. El juego consiste en un tablero de cartas boca abajo y el objetivo es encontrar todas las parejas de cartas iguales.

```

import random
# Función para inicializar el tablero con cartas
def star_card(files, columns):
    cards = list(range(1, (files * columns) // 2 + 1))
    board = [[0] * columns for _ in range(files)]

    for card in cards:
        for _ in range(2):
            while True:
                file = random.randint(0, files - 1)
                column = random.randint(0, columns - 1)
                if board[file][column] == 0:
                    board[file][column] = card
                    break
    return board

# Función para mostrar el tablero con las cartas
def show_board(board, files, columns):
    for file in range(files):
        for column in range(columns):
            if board[file][column] == 0:
                print('?', end=' ')
            else:
                print(board[file][column], end=' ')
        print()

# Función para jugar al Memotest
def play_memotest(files, columns):
    board = star_card(files, columns)
    into_cards = 0
    tries = 0
    while find_cards < files * columns // 2:
        show_board(board, files, columns)
        card1_file = int(input("Ingresa la fila de la primera carta: ")) - 1
        card1_column = int(input("Ingresa la columna de la primera carta: ")) - 1
        card2_file = int(input("Ingresa la fila de la segunda carta: ")) - 1
        card2_column = int(input("Ingresa la columna de la segunda carta: ")) - 1
        if (
            0 <= card1_file < files and 0 <= card1_column < columns and
            0 <= card2_file < files and 0 <= card2_column < columns
        ):
            card1 = board[card1_file][card1_column]
            card2 = board[card2_file][card2_column]

            if card1 == card2 and (card1_file != card2_file or card1_column != card2_column):
                print("¡Encontraste una pareja!")
                find_cards += 1
                board[card1_file][card1_column] = 0
                board[card2_file][card2_column] = 0
            else:
                print("No es una pareja válida.")
            else:
                print("Posición fuera del rango.")
            tries += 1
        show_board(board, files, columns)
        print(f"¡Juego completado en {tries} intentos!")

# Configuración del tamaño del tablero
files = 4
columns = 4
# Jugar al Memotest
play_memotest(files, columns)

```

10. Teniendo una matriz cuadrada de cualquier dimensión, obtener lo siguiente:

- a. la diagonal principal.
- b. la diagonal inversa.

```

# Función para obtener la diagonal principal y diagonal inversa de una matriz cuadrada
def obtain_diagonals(matriz):
    dimension = len(matriz)
    main_diagonal = []
    inverse_diagonal = []
    for i in range(dimension):
        main_diagonal.append(matriz[i][i])
        inverse_diagonal.append(matriz[i][dimension - 1 - i])
    return main_diagonal, inverse_diagonal

# Ejemplo de una matriz cuadrada
square_matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Obtener las diagonales de la matriz
main_diagonal, inverse_diagonal = obtain_diagonals(square_matrix)

# Imprimir las diagonales
print("Diagonal Principal:", main_diagonal)
print("Diagonal Inversa:", inverse_diagonal)

```

11. Escribir un programa que guarde en una variable el diccionario {'Euro':'€', 'Dollar':'\$', 'Yen':'¥'}, pregunte al usuario por una divisa y muestre su símbolo o un mensaje de aviso si la divisa no está en el diccionario.

```

# Crear el diccionario de divisas y símbolos
currencies = {'Euro': '€', 'Dollar': '$', 'Yen': '¥'}

# Solicitar una divisa al usuario
currency = input("Ingresa una divisa (en inglés): ")

# Verificar si la divisa está en el diccionario
if currency in currencies:
    symbol = currencies[currency]
    print(f"El símbolo de {currency} es {symbol}.")
else:
    print(f"La divisa {currency} no está en el diccionario.")

```

12. Escribir un programa que pregunte al usuario su nombre, edad, dirección y teléfono y lo guarde en un diccionario. Después debe mostrar por pantalla el mensaje '<nombre> tiene <edad> años, vive en <dirección> y su número de teléfono es <teléfono>'.

```

1 # Solicitar al usuario su nombre, edad, dirección y teléfono
2 name = input("Ingresa tu nombre: ")
3 age = input("Ingresa tu edad: ")
4 address = input("Ingresa tu dirección: ")
5 phone = input("Ingresa tu número de teléfono: ")
6 # Crear un diccionario con la información ingresada
7 user_information = {
8     'nombre': name,
9     'edad': age,
10    'direccion': address,
11    'telefono': phone
12 }
13
14 # Mostrar un mensaje con los detalles del usuario
15 message = f"{user_information['nombre']} tiene {user_information['edad']} años,
16 vive en {user_information['direccion']} y su número de teléfono es {user_information['telefono']}."
17
18 print(message)

```

13. Escribir un programa que guarde en un diccionario los precios de las frutas de la tabla, pregunte al usuario por una fruta, un número de kilos y muestre por pantalla el precio de ese número de kilos de fruta. Si la fruta no está en el diccionario debe mostrar un mensaje informando de ello.

```
Users > nty Lacoste > OneDrive > Escritorio > x-46.py > ...
1 # Crear un diccionario con los precios de las frutas
2 prices_fruits = {
3     'manzana': 2.5,
4     'banana': 1.5,
5     'naranja': 2.0,
6     'uva': 3.0,
7     'pera': 2.8
8 }
9 # Solicitar al usuario una fruta
10 fruit = input("Ingresa el nombre de la fruta que deseas comprar: ").lower()
11 # Convertir a minúsculas para ser insensible a mayúsculas
12 # Verificar si la fruta está en el diccionario
13 if fruit in prices_fruits:
14     # Solicitar al usuario la cantidad de kilos
15     kilos = input(f"¿Cuántos kilos de {fruit} deseas comprar? ")
16     # Verificar si la cantidad de kilos es un número válido
17     if kilos.replace(".", "", 1).isdigit(): # Comprobar si la cadena es un número
18         kilos = float(kilos)
19         # Calcular el precio total
20         unit_price = prices_fruits[fruit]
21         total_price = unit_price * kilos
22         # Mostrar el precio total
23         print(f"El precio de {kilos} kilos de {fruit} es ${total_price:.2f}")
24     else:
25         print("La cantidad de kilos debe ser un número válido.")
26 else:
27     print(f"Lo siento, la fruta {fruit} no está en la lista de precios.")
28
```