Práctica 1&2

Instrucciones:

Se podrá utilizar un "`Jupiter Notebook`" y crear cada ejercicio en un bloque de código con un comentario que enuncie el número del ejercicio. Se deberá subir al aula virtual el "`.ipynb`" junto con un fichero "`readme`" donde se deberán especificar al menos 7 preguntas realizadas a una LLM (se debe nombrar cual o cuales se han utilizado), si es correcta o no la respuesta y un **mini-resumen** de la respuesta (NO se debe escribir lo que proporciona la LLM, si no vuestras conclusiones). Recuerde que 2 de las 7 preguntas se deben realizar utilizando un tipo de enunciado de pregunta distinto (por ejemplo, utilizando sentimiento/orden…) y ver si la respuesta varía.

Todas las funciones deberán contener un `docstring` y seguir la convención de nombres explicado en clase de teoría.

Nombre los ficheros de la siguiente forma: Nombre_Apellidos_practica1-2

Objetivos:

Introducción al lenguaje de programación Python: bucles, funciones y tipos de datos básicos.

Ejercicio 1

Escriba un programa que genere una tabla de multiplicar y realice las siguientes operaciones:

- Imprimir la tabla de multiplicar para los números del 1 al 5.
- Utilice `continue` para omitir la multiplicación de los números pares.
- Utilice `break` para detener la tabla si el producto supera 20.
- Utilice `pass` dentro del bucle para ilustrar su uso sin afectar la lógica.

Ejercicio 2

Escriba una función llamada `process_text()` que recibe una cadena de texto y una lista de palabras a remplazar. Esta función deberá:

- Proporcionar valores predeterminados para la cadena de texto y la lista de palabras que se reemplazarán.
- Convertir la cadena a minúsculas.
- Eliminar los espacios en blanco iniciales y finales.
- Reemplazar con asteriscos (*) cualquier palabra de la cadena de texto que coincida con un elemento de la lista de palabras.
- Devolver la cadena procesada y la cantidad de palabras reemplazadas.

Llame a `process_text()` desde una función principal.

```
# Ejemplo de salida con valores por defecto. Remplazar: Python amazing

Processed text: "this is a default text with ****** and *******
words."

Palabras remplazadas: 2
```

Ejercicio 3

Escriba un programa que realice un análisis de números primos. Debe incluir las siguientes funciones (que deberán estar documentadas):

1. `is_prime(n)`: comprueba si un número n es primo.
2. `prime_list(limit)`: genera una lista de números primos hasta un límite `limit`.
3. `check_palindrome(primes)`: busca palíndromos en una lista de números primos (se leen igual de adelante hacia atrás).
4. `categorize_prime(prime)`: clasifica los números primos en "pequeños" (menores que 10), "medianos" (de 10 a 99) o "grandes" (100 o más).

La función `main()` se utiliza para probar las funciones del programa. Puedes probar con diferentes números y límites.

Ejercicio 4

Escribe un programa que lea una lista de números enteros y realice las siguientes operaciones utilizando un menú que se repetirá tantas veces como sea necesario hasta que el usuario presione la opción de “salir”. Defina funciones para cada opción y utilice una función `main()`.

1. Calcula la suma de todos los números.
2. Calcula la suma de los números positivos y la suma de los números negativos por separado.
3. Encuentra el número máximo y el mínimo de la lista.
4. Comprueba si todos los números de la lista son únicos.
5. Utiliza el operador condicional para devolver un mensaje que indique si la lista contiene más números positivos o negativos.

Ejercicio 5

Cree un programa que administre una lista de tareas pendientes utilizando variables globales, locales y no locales, y demuestre el uso de funciones integradas (o funciones dentro de otras funciones). Incluya una función `main()`. Como variable global declare la lista de tareas.

Funciones que deberá implementar:

- `add_task(task)`: agrega una nueva tarea a la lista global de tareas.
- `remove_task(task)`: elimina una tarea de la lista global de tareas, si existe.
- `list_tasks()`: enumera todas las tareas almacenadas en la lista global de tareas.
- `task_manager()`: administra las operaciones de agregar, modificar y enumerar tareas.
- `modify_task()`: modifica la primera tarea en la lista de tareas. Esta función está definida dentro de la función `task_manager()`.

Un ejemplo de salida podría ser:

```
Tarea 'Comprar leche' agregada.

Tarea 'Llamar al doctor' agregada.

Tarea 'Hacer ejercicio' agregada.

Lista de tareas:

1. Comprar leche

2. Llamar al doctor

3. Hacer ejercicio

Tarea 'Comprar leche' modificada a 'Comprar leche (modificada)'.

Lista de tareas:

1. Comprar leche (modificada)

2. Llamar al doctor

3. Hacer ejercicio

Tarea 'Hacer ejercicio' eliminada.

Lista de tareas:

1. Comprar leche (modificada)

2. Llamar al doctor
```

Ejercicio 6

Escriba un programa que lea una lista de cadenas desde teclado y realice las siguientes operaciones utilizando un menú que se repetirá tantas veces como sea necesario hasta que el usuario presione la opción de “salir”. Defina funciones para cada opción y utilice una función `main()`.

1. Imprima cada cadena que comience con una letra pedida al usuario.
2. Cuente cuántas cadenas contienen una subcadena pedida al usuario.
3. Encuentre la cadena más larga y la más corta de la lista.
4. Utilice el operador `is` para verificar si dos cadenas específicas de la lista son en realidad el mismo objeto.
5. Utilice un bucle `for...else` para verificar si alguna cadena de la lista tiene una longitud mayor que 10. Si no es así, imprima un mensaje que indique que ninguna cadena tiene más de 10 caracteres.