**ESCUELA POLITÉCNICA NACIONAL**

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

**PROGRAMACIÓN**

**(TDSD214)**

|  |  |
| --- | --- |
|  | |
| ASIGNATURA: | Programación |
| PROFESOR: | Ing. Lorena Chulde |
| PERÍODO ACADÉMICO: | 2023-B |
|  | |

**DEBER**

|  |
| --- |
|  |
| **TÍTULO**  **ARREGLOS**  **FUNCIONES – PASO DE PARÁMETROS POR TUPLA, LISTAS**  **Nombres de los estudiantes:**  **Guerra Lovato Josué**  **Pérez Orosco Carlos**  **Soria Ansa Richard** |
|  |
|  |
| Tuplas - Aprende Python |
|  |

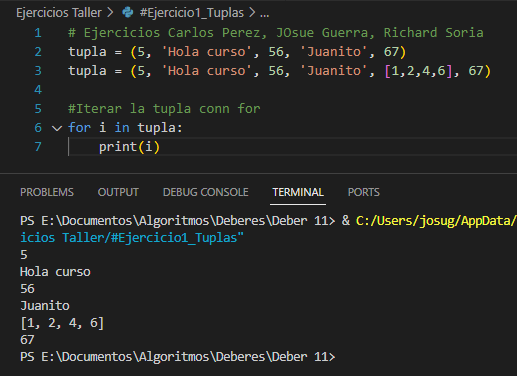
**2023-B**

**PROPÓSITO DE LA TAREA**

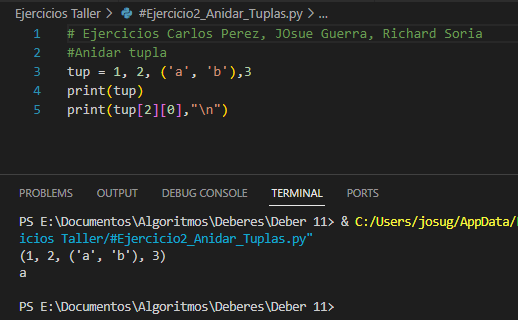
Recorrer tuplas mediante estructuras cíclicas para leerlos y visualizar sus valores en pantalla.

**TALLER**

1. Iterar una tupla



1. Anidar tuplas



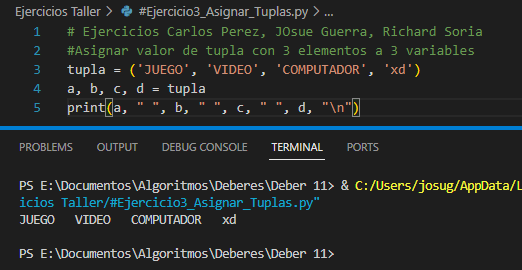
* **En el código proporcionado, se define una tupla llamada tup que contiene cuatro elementos: los números enteros 1, 2 y 3, y una tupla anidada que contiene los caracteres 'a' y 'b'. Al imprimir tup, se muestra el contenido completo de la tupla. Luego, al acceder a tup[2][0], se extrae el primer elemento de la tupla anidada, que es 'a', y se imprime. En resumen, el código crea una estructura de datos inmutable que combina diferentes tipos de elementos, y permite acceder a elementos individuales dentro de la tupla anidada mediante indexación.**

1. Asignar el valor de una tupla con n elementos a n variables.

## Métodos tuplas

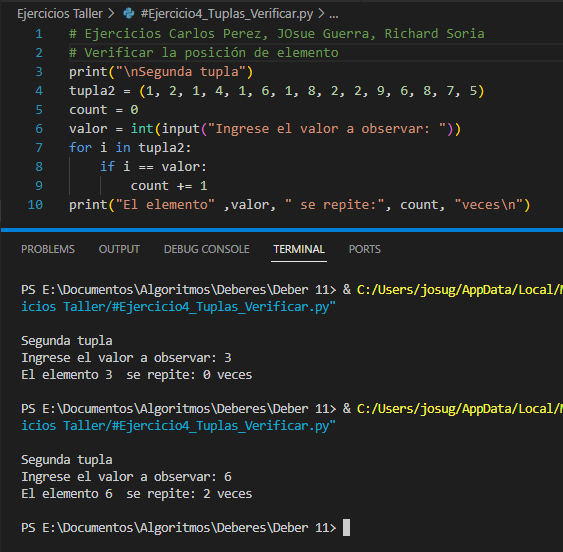
### count(<obj>)

### index(<obj>[,index])

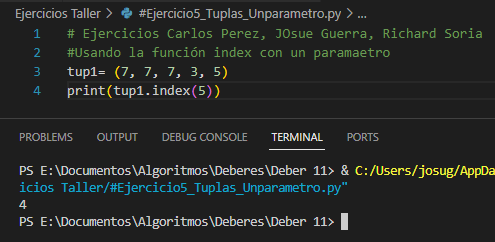


Como tal dentro del Código se asigna un valor a la tuple y como se realize en clase el valor se que com 3 elementos y 3 variables.

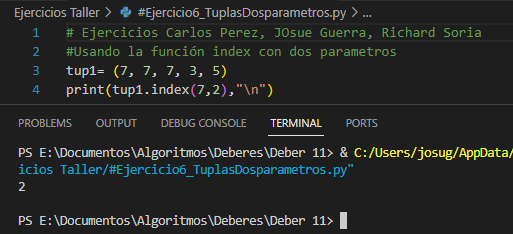
1. Verificar la posición de un elemento:



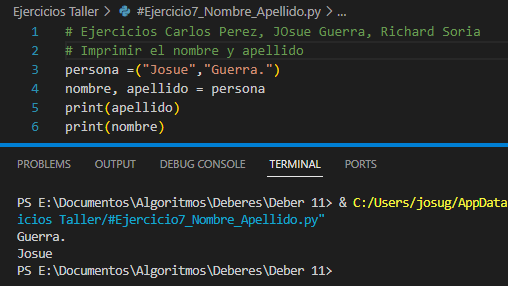
1. Con un parámetro



1. Con dos parámetros

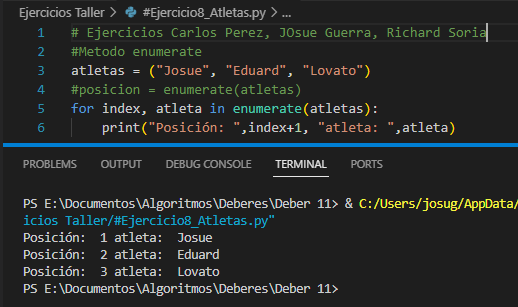


1. Imprimir el apellido y el nombre

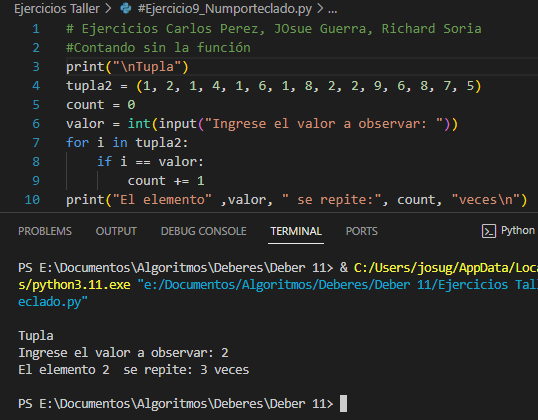


1. Imprimir el orden de llegada de los atletas

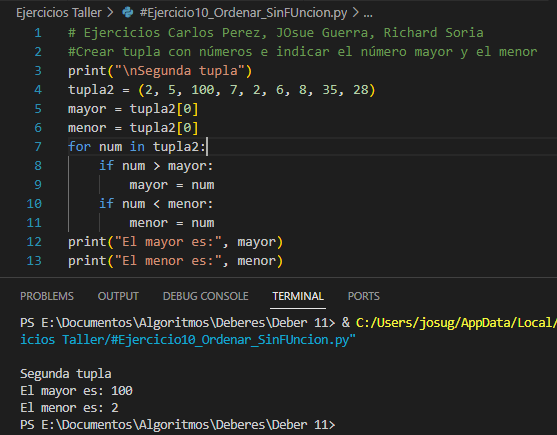
* Verificar que es un objeto enumerate



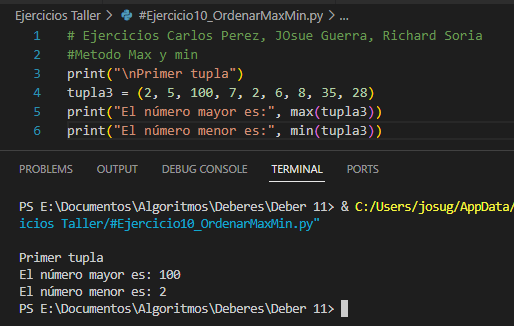
1. Crea una tupla con números, pide un numero por teclado e indica cuantas veces se repite.



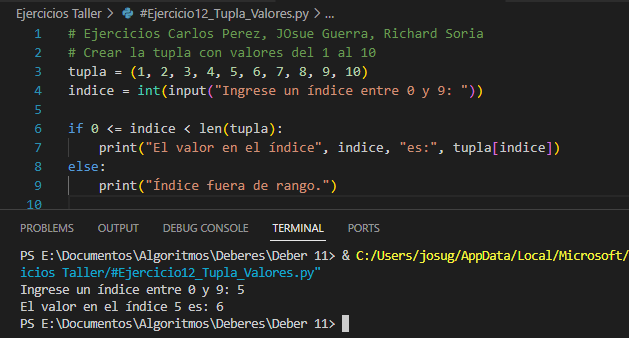
1. Crea una tupla con números e indica el numero con mayor valor y el que menor tenga.

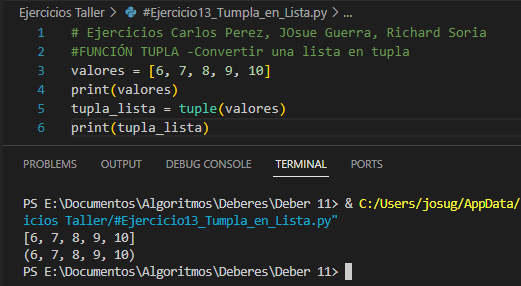


1. Usando max, min



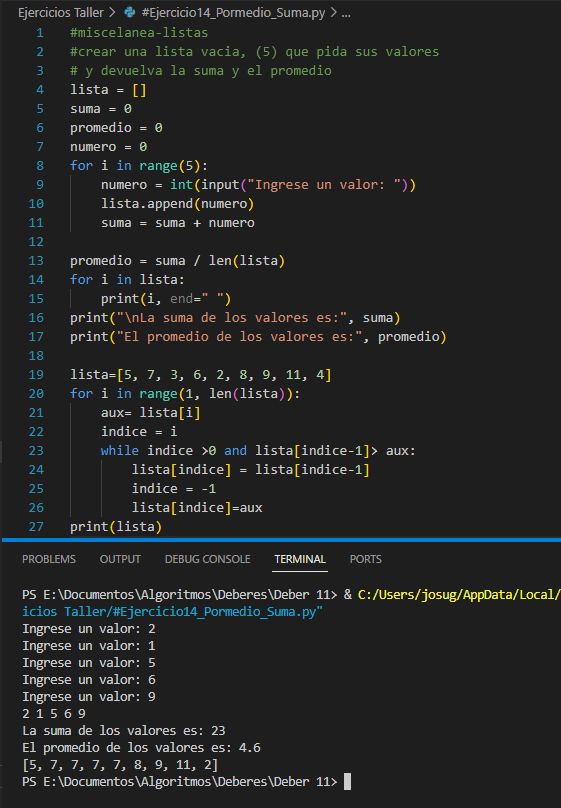
La función MAX y MIN realiza el orden de forma automática y eso lo imprime en pantalla, lo cual no es tan necesario que se haga todo el código para poder ordenar valores.

1. Crea una tupla con valores ya predefinidos del 1 al 10, pide un índice por teclado y muestra el valor de la tupla.
2. Convertir una lista en tupla haciendo uso de al función tuple().



Con la función tuple la lista se convierte automáticamente en una tupla, lo que nos facilitaría para la creación de códigos.

1. Crea una lista vacía (pongamos 10 posiciones), pide sus valores y devuelve la suma y la media de los números.



**Algoritmos de ordenamiento por inserción**

1. Realizar la prueba de escritorio

**Forma corta**

lista = [5,7,1,3,8,4,9,2,6]

for i in range(1, len(lista)):

    actual = lista[i]

    indice = i

    while indice> 0 and lista [indice-1] >actual:

        lista[indice] = lista[indice-1]

        indice = indice-1

    lista[indice] = actual

print(list)

***Prueba de escritorio de forma escrita***

**Iteración 0:**

La lista es [5, 7, 1, 3, 8, 4, 9, 2, 6].

No hay ningún valor actual ni índice en esta etapa.

La lista no cambia en esta etapa.

Iteración 1:

El valor actual es 7.

El índice es 1.

Como 7 es mayor que 5, no hay cambios en la lista.

La lista sigue siendo [5, 7, 1, 3, 8, 4, 9, 2, 6].

**Iteración 2:**

El valor actual es 1.

El índice es 2.

Comparamos 1 con 7 y 5. 1 es menor que ambos.

Desplazamos 7 y 5 una posición a la derecha.

Actualizamos el índice a 0.

La lista se convierte en [1, 5, 7, 3, 8, 4, 9, 2, 6].

**Iteración 3:**

El valor actual es 3.

El índice es 3.

Comparamos 3 con 7, 5 y 1. 3 es menor que 7 pero mayor que 1.

Desplazamos 7 y 5 una posición a la derecha.

Colocamos 3 en la posición correcta.

La lista se convierte en [1, 3, 5, 7, 8, 4, 9, 2, 6].

**Iteración 4:**

El valor actual es 8.

El índice es 4.

Como 8 es mayor que 7, no hay cambios en la lista.

La lista sigue siendo [1, 3, 5, 7, 8, 4, 9, 2, 6].

**Iteración 5:**

El valor actual es 4.

El índice es 5.

Comparamos 4 con 8, 7, 5 y 3. 4 es menor que todos.

Desplazamos 8, 7 y 5 una posición a la derecha.

Colocamos 4 en la posición correcta.

La lista se convierte en [1, 3, 4, 5, 7, 8, 9, 2, 6].

**Iteración 6:**

El valor actual es 9.

El índice es 6.

Como 9 es mayor que 8, no hay cambios en la lista.

La lista sigue siendo [1, 3, 4, 5, 7, 8, 9, 2, 6].

**Iteración 7:**

El valor actual es 2.

El índice es 7.

Comparamos 2 con 9, 8, 7, 5 y 4. 2 es menor que todos.

Desplazamos 9, 8, 7, 5 y 4 una posición a la derecha.

Colocamos 2 en la posición correcta.

La lista se convierte en [1, 2, 3, 4, 5, 7, 8, 9, 6].

**Iteración 8:**

El valor actual es 6.

El índice es 8.

Comparamos 6 con 9, 8 y 7. 6 es menor que 9 pero mayor que 2.

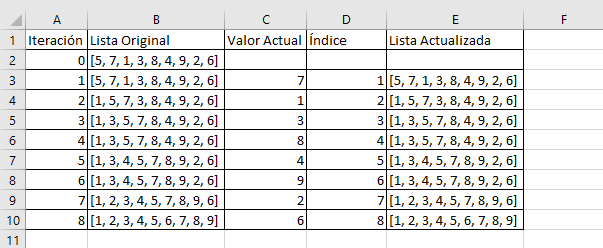
Desplazamos 9, 8 y 7 una posición a la derecha.

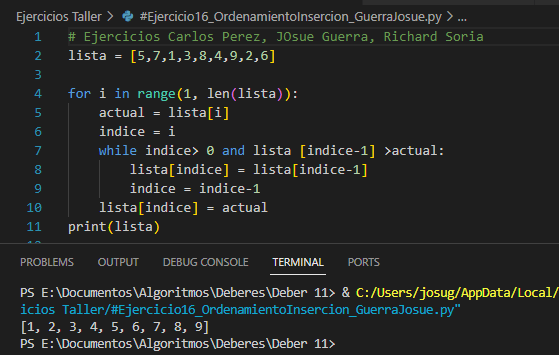
Colocamos 6 en la posición correcta.

La lista se convierte en [1, 2, 3, 4, 5, 6, 7, 8, 9].

* Después de la última iteración, la lista está ordenada correctamente. Por lo tanto, el resultado final es [1, 2, 3, 4, 5, 6, 7, 8, 9].

***Prueba de escritorio en formato de tabla:***

******

******

***Ejecución del código:***

***Como se puede observar el código mostrara la lista de forma ordenada acomodando cada valor desde el menor hasta el mayor.***

1. Realizar la prueba de escritorio

**Función para mostrar estado de la lista**

def mostrarLista(lista, lon):

    listaordenada=""

    for i in range(0,lon):

        listaordenada+=str(lista[i])+" "

    print(listaordenada)

arreglo = [5,2,4,1,3];

#Recorrer el arreglo

for i in range(1,len(arreglo)):

    clave = arreglo[i]

    j = i-1

    #Comparar el valor seleccionado con todos los valores anteriores

    while (j>=0 and arreglo[j] > clave):

        #Insertar el valor donde corresponda

        arreglo[j+1] = arreglo[j]

        j = j-1

    arreglo[j+1] = clave

    mostrarLista(arreglo, len(arreglo))

mostrarLista(arreglo, len(arreglo))

**Prueba de escritorio de forma escrita:**

Se define la función mostrarLista(lista, lon) que toma dos argumentos: lista, que es la lista que se mostrará, y lon, que es la longitud de la lista.

Se define el arreglo [5,2,4,1,3].

Se ejecuta un bucle for que itera sobre los índices del arreglo, comenzando desde 1 hasta la longitud del arreglo (len(arreglo)).

En cada iteración, se toma el elemento en la posición i y se almacena en la variable clave.

Se inicializa j con el valor i-1.

Se ejecuta un bucle while que compara el valor de arreglo[j] con clave y mueve los elementos mayores hacia la derecha en la lista.

Una vez que se encuentra la posición correcta para insertar clave, se inserta en arreglo[j+1].

Se llama a la función mostrarLista() para mostrar el estado actual del arreglo en esa iteración.

Después del bucle for, se llama a la función mostrarLista() una vez más para mostrar el estado final del arreglo.

**Primera iteración (i = 1):**

arreglo es [2,5,4,1,3].

**Segunda iteración (i = 2):**

arreglo es [2,4,5,1,3].

**Tercera iteración (i = 3):**

arreglo es [1,2,4,5,3].

**Cuarta iteración (i = 4):**

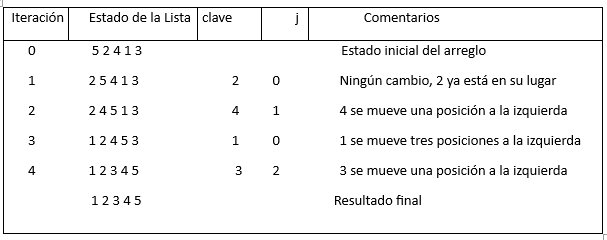
arreglo es [1,2,3,4,5].

**Quinta iteración (i = 5):**

arreglo sigue siendo [1,2,3,4,5].

* El resultado final es [1,2,3,4,5]. Cada línea impresa en el proceso muestra el estado actual del arreglo en cada iteración.

**Prueba de escritorio en formato de tabla:**



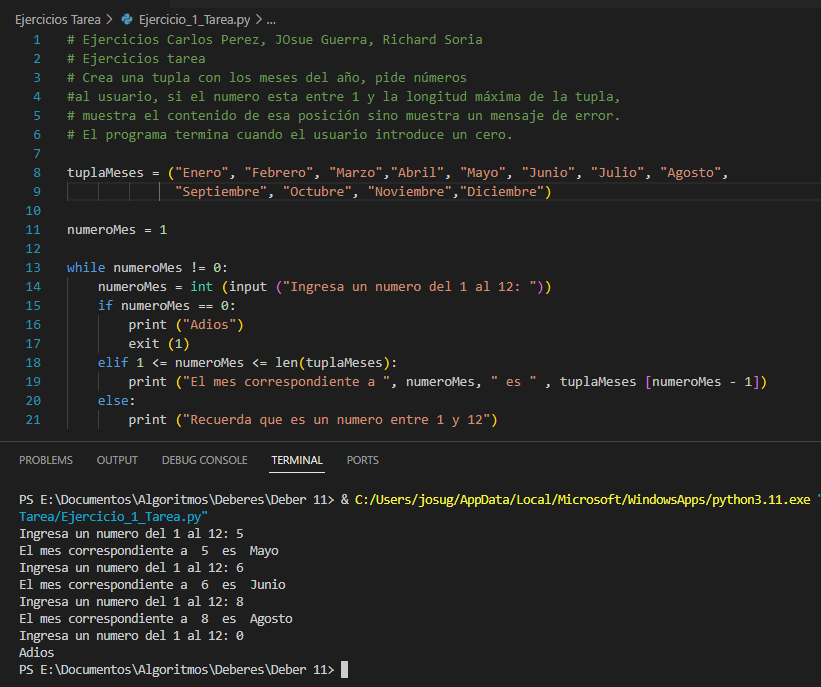
**Ejecución del código:**

**El código implementa el algoritmo de ordenamiento de inserción en Python. Comienza con una lista desordenada arreglo, y luego itera sobre cada elemento de la lista, comenzando desde el segundo elemento (i = 1). Dentro del bucle for, el código selecciona un elemento como la "clave" y lo compara con los elementos anteriores en la lista, desplazando los elementos mayores hacia la derecha para hacer espacio para la inserción de la clave en su posición correcta. Finalmente, se llama a la función mostrarLista() para imprimir el estado actualizado de la lista en cada iteración del algoritmo. Una vez que el bucle ha terminado, se llama a mostrarLista() nuevamente para imprimir la lista completamente ordenada.**

**TAREA**

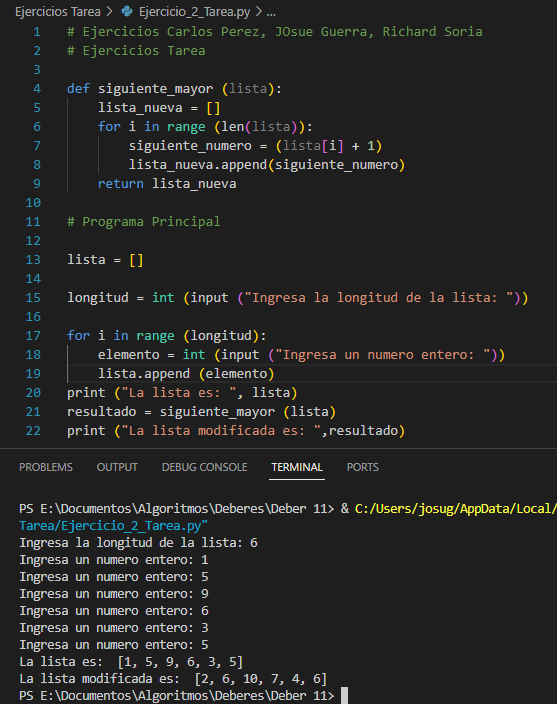
1. Crea una tupla con los meses del año, pide números al usuario, si el numero esta entre 1 y la longitud máxima de la tupla, muestra el contenido de esa posición sino muestra un mensaje de error.

El programa termina cuando el usuario introduce un cero.



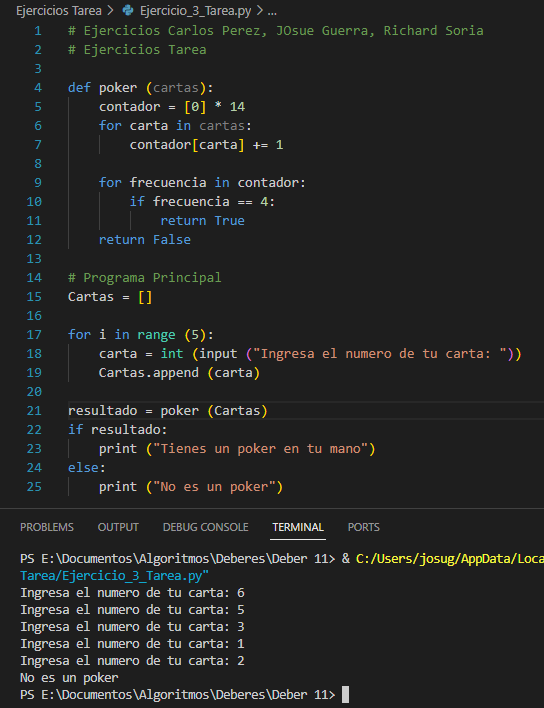
**El código presenta un bucle interactivo que solicita al usuario ingresar un número del 1 al 12, correspondiente a un mes del año. Si el usuario ingresa 0, el programa finaliza con el mensaje "Adios". Si el número ingresado está dentro del rango válido (1 al 12), se imprime el nombre del mes correspondiente utilizando la tupla tuplaMeses. Si el número ingresado está fuera de ese rango, se muestra un mensaje recordando al usuario el rango permitido. La ejecución continúa hasta que el usuario ingresa 0 para salir del programa. La tupla tuplaMeses contiene los nombres de los meses en orden, y se utiliza para obtener el nombre del mes correspondiente al número ingresado por el usuario. La función exit(1) se utiliza para finalizar el programa con un código de salida 1.**

1. Implementa una función siguiente\_mayor(lista), a la que se le pase como argumento una lista de números enteros y reemplace cada número por el siguiente más grande de la lista.

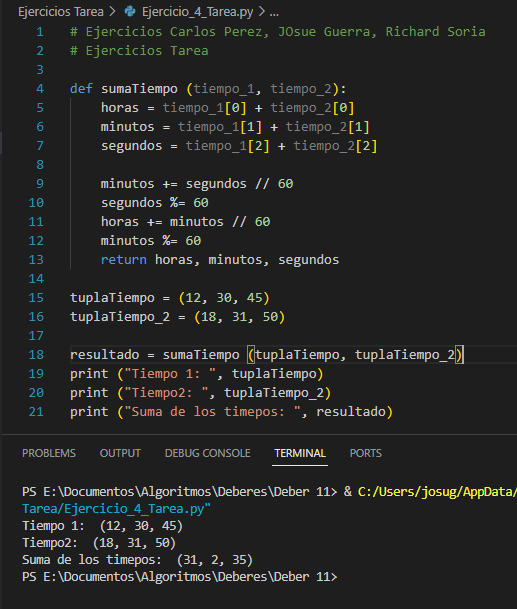


**El código define una función llamada siguiente\_mayor() que toma una lista de números como entrada y devuelve una nueva lista donde cada elemento es el número siguiente al correspondiente en la lista original. Luego, en el programa principal, se solicita al usuario ingresar la longitud de la lista y los elementos de la lista. Posteriormente, se llama a la función siguiente\_mayor() pasando la lista ingresada como argumento, y se imprime la lista original y la lista modificada. En resumen, el programa toma una lista de números y devuelve una lista donde cada número ha sido incrementado en 1.**

1. Escribir una función poker que reciba cinco cartas de la baraja francesa e informe (devuelva el valor lógico correspondiente) si esas cartas forman o no un poker (es decir que hay 4 cartas con el mismo número).

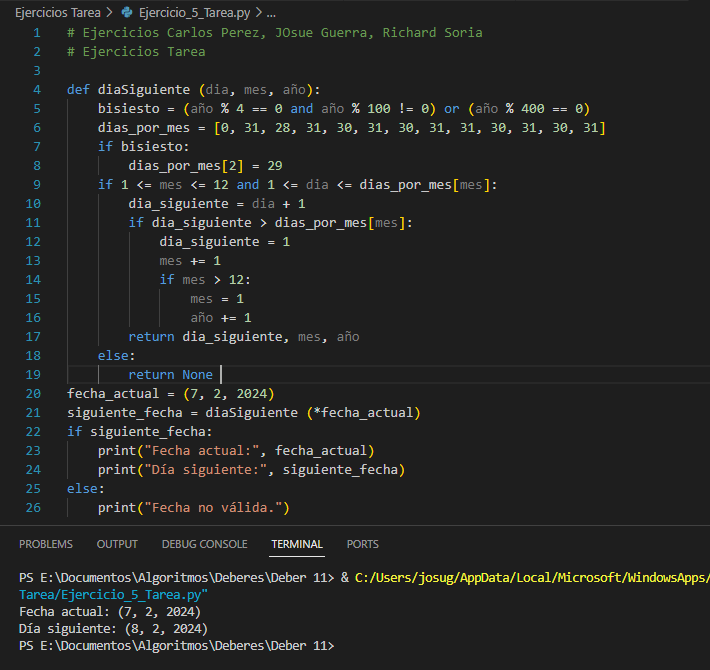


1. El tiempo como tuplas. Proponer una representación con tuplas para representar el tiempo. Escribir una función sumaTiempo que reciba dos tiempos dados y devuelva su suma.



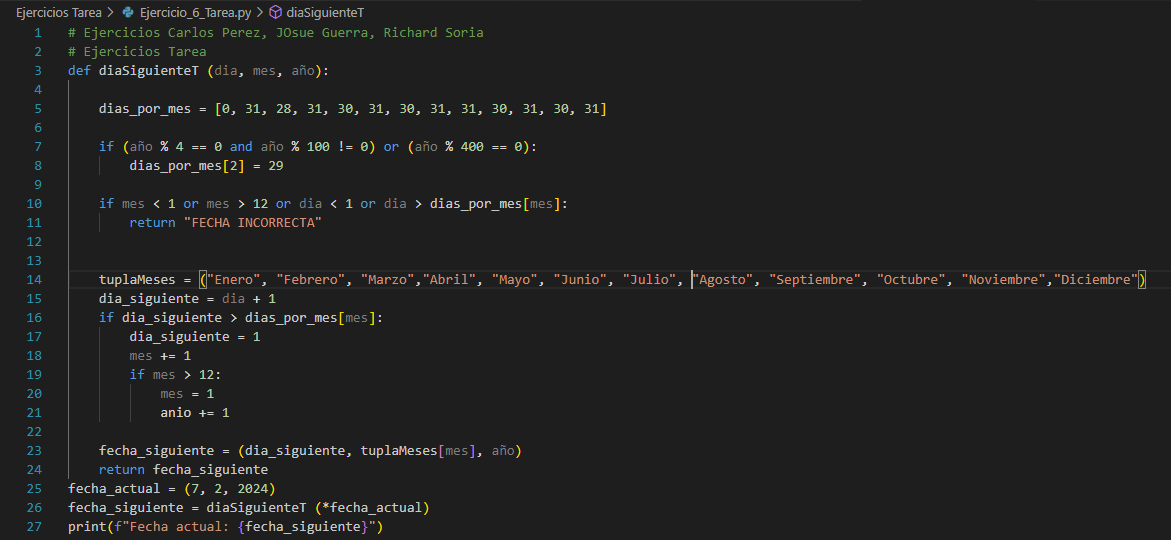
**El código define una función llamada sumaTiempo que toma dos tuplas como argumentos, cada una representando una cantidad de tiempo en horas, minutos y segundos. La función suma los tiempos proporcionados y devuelve una nueva tupla que representa la suma de ambos tiempos. Luego, en el programa principal, se definen dos tuplas tuplaTiempo y tuplaTiempo\_2, que representan dos periodos de tiempo. Se llama a la función sumaTiempo con estas dos tuplas como argumentos, y se imprime el resultado junto con las dos tuplas originales. En resumen, el programa calcula la suma de dos periodos de tiempo representados por tuplas y muestra el resultado junto con los tiempos originales.**

1. Escribir una función diaSiguienteE que dada una fecha expresada como la terna (Día, Mes, Año) (donde Día, Mes y Año son números enteros) calcule el día siguiente al dado, en el mismo formato.

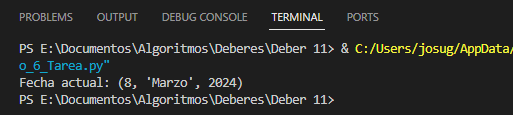


**El código implementa la función diaSiguiente, que calcula la fecha del día siguiente a partir de una fecha dada, considerando años bisiestos. Si la fecha proporcionada es válida, se imprime la fecha actual y la siguiente fecha. De lo contrario, se imprime un mensaje de "Fecha no válida". La función ajusta correctamente los días, meses y años cuando se alcanzan los límites de un mes o un año. En resumen, el programa realiza un cálculo preciso de la fecha del día siguiente y maneja adecuadamente los casos de fechas inválidas.**

1. Escribir una función diaSiguienteT que dada una fecha expresada como la terna (Día, Mes, Año) (donde Día y Año son números enteros, y Mes es el texto Ene, Feb, ..., Dic, según corresponda) calcule el día siguiente al dado, en el mismo formato.



Ejecución del código:



**El código define la función diaSiguienteT para calcular la fecha del día siguiente a partir de una fecha dada, considerando años bisiestos. Verifica la validez de la fecha y ajusta correctamente los días y meses si es necesario. Retorna "FECHA INCORRECTA" si la fecha dada no es válida. En el programa principal, se proporciona una fecha actual y se imprime la fecha del día siguiente obtenida con la función. Este código ofrece una solución precisa y eficiente para calcular la fecha siguiente, teniendo en cuenta diversas condiciones y verificando la validez de la fecha proporcionada.**

El siguiente ejercicio se encuentra en la siguiente página.

****

1. **“Rapid”** es una empresa que entrega artículos de bazar como regalos, peluches, chocolates, perfumes, flores, cartas, etc. Su tienda permite al cliente armar un paquete y enviarlo a la dirección del destinatario,

|  |  |
| --- | --- |
| **DETALLE** | **COSTO** |
| Flores + peluche | $15,50 |
| Flores + carta | $7,50 |
| Flores + Chocolates | $12,25 |
| Flores + perfume | $22,75 |

Usted es contratado para automatizar la tienda de RAPID.

1. Deberá gestionar los pedidos (CRUD) de los clientes presentándoles el siguiente menú:

**\*\*\*\*\*\*\*\*\*RAPID \*\*\*\*\*\*\*\*\*\***

**Qué acción desea realizar?**

1. **Registrar compra**
2. **Mostrar compra**
3. **Mostrar el detalle de la compra**
4. **Salir del sistema**

**Ingrese la opción:**

Si ingresa la **opción 4** debe salir del sistema.

1. Si el administrador ingresa en la **opción 1** el sistema debe permitir:
   1. Registrar nuevas compras.
   2. Si el administrador nuevamente ingresa a la opción 1, se deben registrar nueva compra a los ya existentes.

**\*\*\*\*\*\*\*\*\* RAPID \*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\* Nueva compra \*\*\*\*\***

**Ingrese los datos del cliente**

Nombre:

Apellido:

Telefono

Correo:

**Ingrese los datos de la persona a quién desea enviar el regalo:**

Nombre:

Dirección:

Teléfono:

Seleccione el paquete:

1. Flores + peluche costo $15,50
2. Flores + carta costo $7,50
3. Flores + Chocolates costo $12,25
4. Flores + perfume costo $22,75

Ingrese la opción 4

Compra registrada con éxito

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Qué desea realizar?

1. Registrar compra
2. Mostrar compra
3. Mostrar el detalle de la compra
4. Salir del sistema

**Ingrese la opción:**

* 1. Si selecciona la **opción 2** debe permitir:
     + - 1. Visualizar todas las compras que se han registrado en la opción 1.

**\*\*\*\*\*\*\*\*\* Detalle de la compra \*\*\*\*\*\*\*\*\*\*\***

**Datos del cliente**

Nombre: Juan Pablo

Apellido: Pérez

Telefono: 0986677445

Correo: juan@gamail.com

**Datos de la entrega**

Nombre: María Mena

Dirección: Cumbayá

Teléfono: 097865432

Datos del pago

1. Número de factura
2. Precio final: $30.25

Ingrese la opción 4

Compra registrada con éxito

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

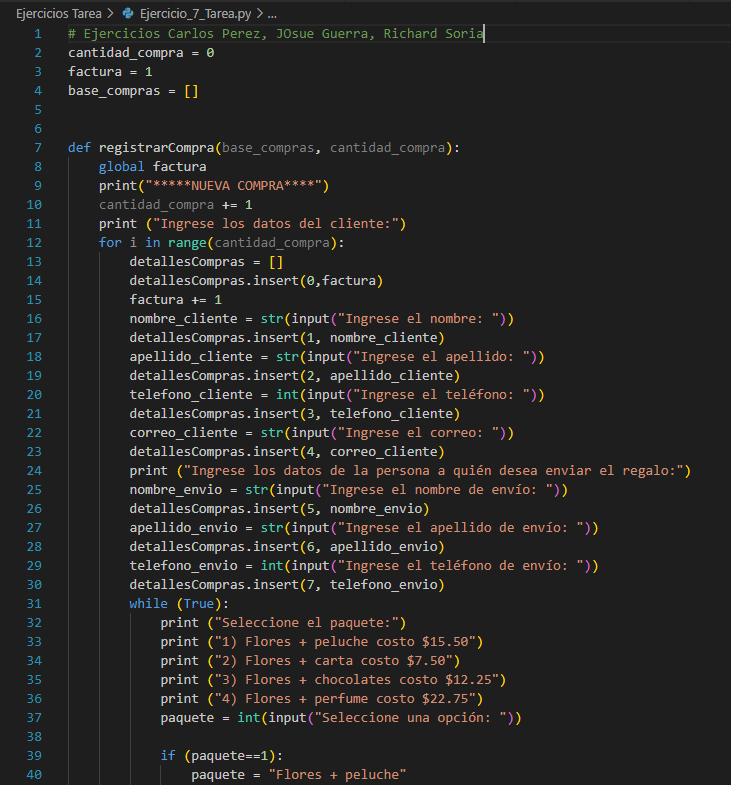
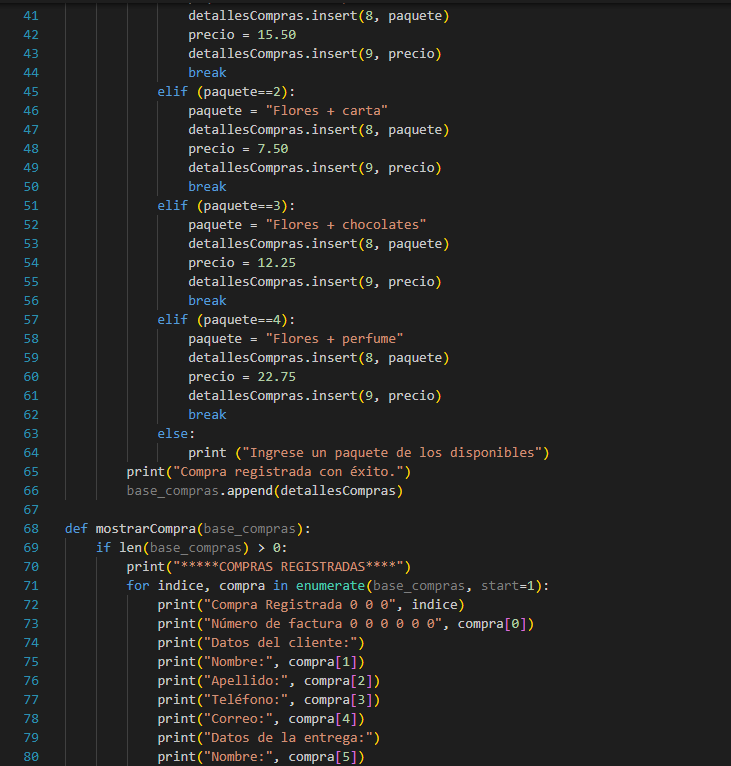
Qué desea realizar?

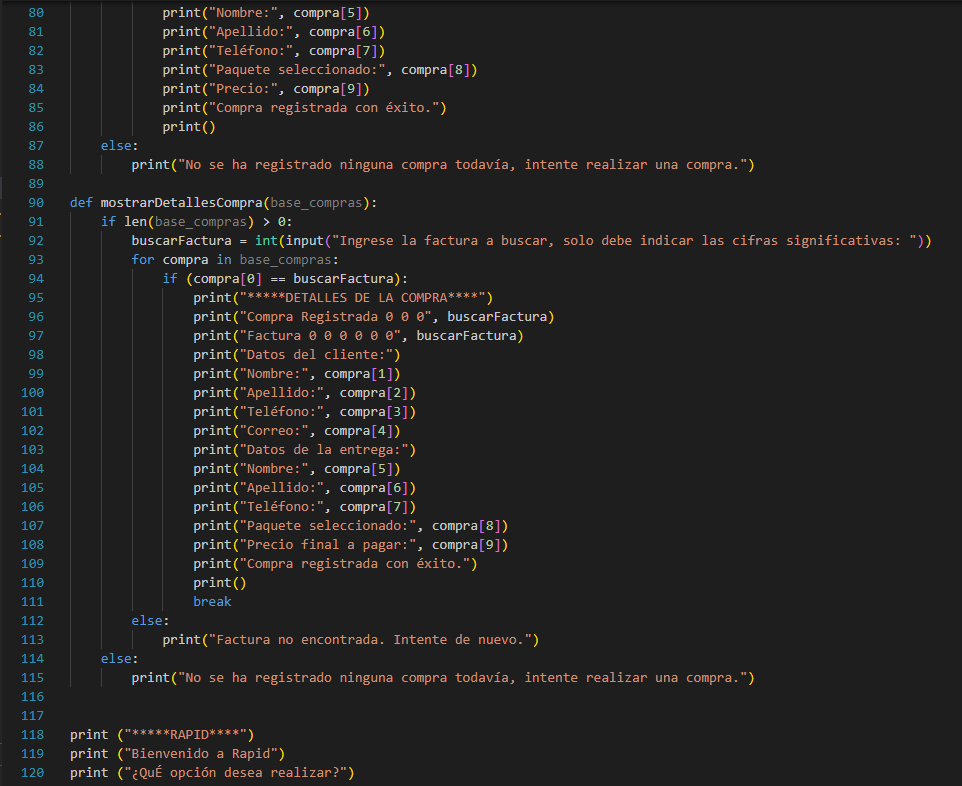
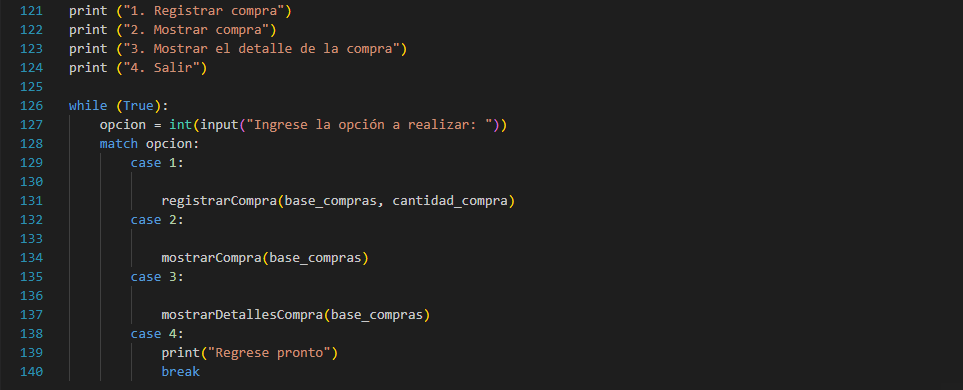
1. Registrar compra
2. Mostrar compra
3. Mostrar el detalle de la compra
4. Salir del sistema

**Ingrese la opción:**

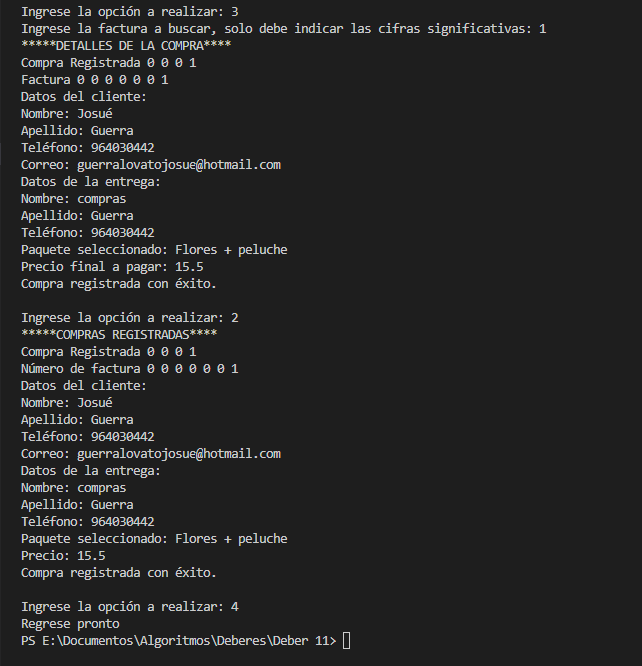
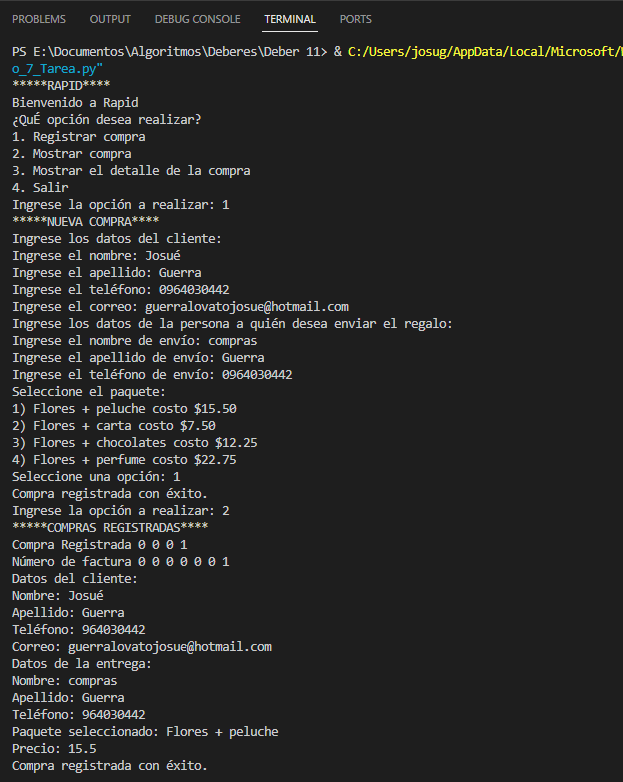
* + - * 1. Y en el caso de que no existan compras registradas, se debe mostrar un mensaje que mencione que no existen compras registradas.
  1. Si selecciona la **opción 3** debe visualizar el detalle de la compra con el número de factura, caso contrario mostrar un mensaje de error.

**El código y su ejecución se encuentran en la siguiente página.**

****

****

**El código simula un sistema de registro de compras en una tienda llamada "Rapid". Ofrece opciones para registrar nuevas compras, mostrar todas las compras registradas y ver los detalles de una compra específica. Las funciones registrarCompra, mostrarCompra, y mostrarDetallesCompra manejan las diferentes operaciones de registro y visualización de compras. Un bucle principal permite al usuario seleccionar entre estas opciones, con una opción para salir del programa. Este código proporciona una estructura completa para gestionar el proceso de compras en la tienda de manera eficiente y organizada.**

**Ejecución del código:**

1. **Consultar sobre archivos**

**¿Qué es el manejo de archivos en Python?**

Para empezar con lo básico, se debe entender qué es un archivo. Un archivo es la información o los datos que se recopilan en los distintos dispositivos de almacenamiento de una computadora. Estos pueden ser de música, video, texto, entre otros.

Cuando se trabaja con Python, en general, se dividen los archivos en dos categorías: texto y binarios. Los archivos de texto son texto simple, mientras que los archivos binarios contienen datos que solo pueden ser interpretados por una computadora. Python ofrece formas simples de manipular ambos tipos de archivos. Esto es lo que se conoce como el manejo de archivos en Python.

Python permite y facilita el manejo de archivos, es decir, proporciona a los programadores la capacidad de leer y escribir en estas piezas de datos. Sin embargo, las opciones disponibles para el manejo de archivos con Python no se limitan a esto, ya que existen otras formas de operar con ellos.

El manejo de archivos es un concepto ampliamente utilizado en programación, pero su implementación puede ser complicada. Sin embargo, Python, al igual que con muchas de sus otras funcionalidades, ha logrado simplificar y abreviar este proceso. Esto se debe, entre otras cosas, a cómo Python maneja los archivos binarios. A continuación, se explorará cómo se manejan los archivos en Python.

**¿Cómo se manejan los archivos en Python?**

Como decíamos, una de las grandes ventajas que ofrece Python es su capacidad para el manejo de archivos, lectura, escritura y otras opciones que veremos a continuación. Primero vamos a empezar por ver como hace que la gestión de archivos sea mucho más sencilla que otros lenguajes de programación.

En Python, cada línea de código incluye una secuencia de caracteres que juntos forman un archivo de texto. Cada una de las líneas de esos archivos termina con un carácter especial denominado EOL o caracteres de final de línea o nueva línea. Con ello, sabemos dónde finaliza y donde comienza cada parte durante la lectura y la escritura de archivos en Python.

**Lectura en el manejo de archivos en Python**

Python ofrece una función con la que se puede crear un objeto de archivo con varios modos, uno de ellos es el modo de lectura.

El modo de lectura en Python permite devolver el contenido del archivo. De esta manera, el contenido del archivo se mostrará una vez que se ejecute el código relacionado con el modo de lectura.

La función de lectura lee todo el archivo a la vez, pero esto se puede cambiar. Se puede especificar un tamaño de byte con los parámetros de implementación del modo lectura.

Por último, es importante destacar que Python incorpora funciones integradas para leer un archivo línea a línea. Esto simplifica el proceso y evita que los programadores tengan que implementar el mecanismo de lectura de archivos.

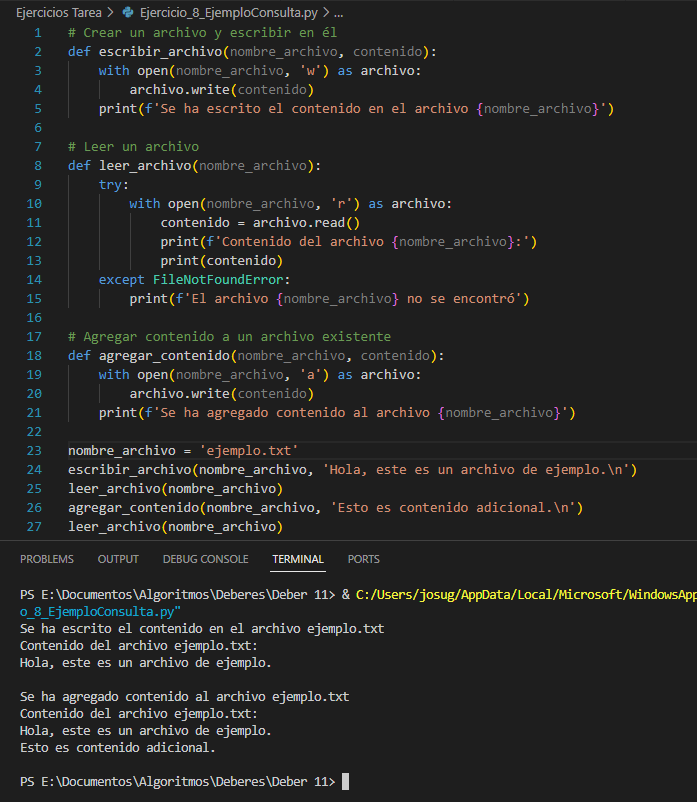
**Escritura de archivos en Python**

El modo o la función de escritura en Python puede habilitarse al mismo tiempo que se ejecuta la función de lectura, o se puede optar por ejecutar ambas de manera independiente, según las necesidades de cada proyecto y programador.

Con la función de escritura, se define el contenido del archivo de texto utilizando la sintaxis de cadenas de Python.

En el caso de que se necesite agregar nuevo contenido a un archivo ya programado mediante la función de escritura, Python permite añadir nuevos contenidos a archivos de texto en el manejo de archivos.

En este caso, aplicar las funciones de lectura y escritura a todo el contenido de nuevo es una opción que consume muchos recursos. Por eso, existe la función de adición, en la cual se pueden añadir los nuevos contenidos para la escritura del archivo [1].

**Ejemplo de uso de archivos en Python:**

**ENTREGABLES:**

Una vez culminada tu tarea, capturar las pantallas de la ejecución del problema con tus datos y súbela en el apartado del aula virtual “S12-Deber-5: Tuplas, listas, funciones”

Subir los ejercicios al git o al drive y entrega la url de los archivos .py o, a su vez, entregue el archivo.

1. Recordar que el nombre del archivo deberá ser: S12-**Deber-5-2023B-NApellido**(de todos los integrantes)

**RECURSOS NECESARIOS**

* Acceso a Internet.
* Imaginación.
* VSC

**RECOMENDACIÓN**

Para optimizar el desarrollo en Python, es crucial dominar el manejo de arreglos, funciones y archivos. Se recomienda practicar regularmente la manipulación de arreglos para familiarizarse con las diferentes operaciones disponibles, como la indexación, el corte y la manipulación de elementos. Además, es importante comprender profundamente cómo funcionan las funciones en Python y cómo pasar parámetros por tuplas y listas para aprovechar al máximo su potencial. Asimismo, se aconseja practicar el manejo de archivos mediante la lectura y escritura de datos en diferentes formatos, como texto y binario. Dominar estos conceptos y técnicas permitirá a los desarrolladores crear aplicaciones más sólidas y eficientes en Python.

**CONCLUSIÓN**

El manejo efectivo de arreglos, funciones y archivos en Python es esencial para el desarrollo de aplicaciones eficientes y robustas. Los arreglos ofrecen una forma conveniente de almacenar y manipular datos de manera estructurada, lo que facilita la gestión de conjuntos de información. Las funciones permiten modularizar el código, promoviendo la reutilización y facilitando el mantenimiento del programa. Además, el paso de parámetros por tuplas y listas en funciones brinda flexibilidad y potencia para trabajar con datos dinámicos y complejos. Por último, el uso de archivos en Python proporciona la capacidad de leer y escribir datos de manera persistente, lo que es fundamental para aplicaciones que requieren almacenamiento de información a largo plazo.

**ENLACES**

**Enlace GitHub:**

[**https://github.com/JosueGuerra2023B/Estructuras-Datos2023B/tree/master/Deberes/Deber%2011**](https://github.com/JosueGuerra2023B/Estructuras-Datos2023B/tree/master/Deberes/Deber%2011)

# Bibliografía

|  |  |
| --- | --- |
| [1] | T. School, «Esteban Canle Fernández,» 3 Mayo 2022. [En línea]. Available: https://www.tokioschool.com/noticias/manejo-archivos-python/#%c2%bfque-es-el-manejo-de-archivos-en-python. [Último acceso: 7 Febrero 2024]. |