

Taller de Manipulación de Listas en Python

Ejercicio 1: Crear una Lista

Crea una lista con los números del 1 al 5 y muestra el resultado.

```
lista = [1, 2, 3, 4, 5]  
  
print(lista)
```

Ejercicio 2: Acceder al Tercer Elemento

Accede y muestra el tercer elemento de la lista creada anteriormente.

```
print(lista[2])
```

Ejercicio 3: Modificar el Segundo Elemento

Cambia el segundo elemento de la lista a 10 y muestra la lista modificada.

```
lista[1] = 10  
  
print(lista)
```

Ejercicio 4: Agregar un Elemento

Usa `append()` para agregar el número 6 al final de la lista y muestra la lista actualizada.

```
lista.append(6)  
  
print(lista)
```

Ejercicio 5: Eliminar un Elemento por Valor

Usa `remove()` para eliminar el número 3 de la lista y muestra el resultado.

```
busca=3  
  
lista.remove(busca)  
  
print(lista)
```

Ejercicio 6: Eliminar un Elemento por Índice

Usa `del` para eliminar el primer elemento de la lista y muestra la lista después de la eliminación.

```
i=0  
del lista[i]  
print(lista)
```

Ejercicio 7: Longitud de la Lista

Muestra la longitud de la lista actual.

```
print(len(lista))
```

Ejercicio 8: Extender la Lista

Usa `extend()` para agregar los números [7, 8, 9] al final de la lista y muestra el resultado.

```
lista.extend([7, 8, 9])  
print(lista)
```

Ejercicio 9: Insertar un Elemento

Usa `insert()` para agregar el número 0 al inicio de la lista y muestra la lista modificada.

```
lista.insert(0, 0)  
print(lista)
```

Ejercicio 10: Limpiar la Lista

Usa `clear()` para eliminar todos los elementos de la lista y muestra la lista vacía.

```
lista.clear()  
print(lista)
```

Ejercicio 11: Revertir la Lista

Crea una lista nueva con los números del 1 al 5, revierte su orden con `reverse()` y muestra el resultado.

```
lista = [1, 2, 3, 4, 5]
```

```
lista.reverse()
```

```
print(lista)
```

Ejercicio 12: Ordenar la Lista

Crea una lista desordenada, úsala para ordenar los elementos con `sort()` y muestra la lista ordenada.

```
lista_desordenada = [5, 3, 1, 4, 2]
```

```
lista_desordenada.sort()
```

```
print(lista_desordenada)
```

Ejercicio 13: Índice de un Elemento

Encuentra y muestra el índice del número 4 en la lista ordenada.

```
indice = lista_desordenada.index(4)
```

```
print(indice)
```

Ejercicio 14: Último Elemento con `pop()`

Muestra y elimina el último elemento de la lista usando `pop()`, y luego muestra la lista modificada.

```
ultimo = lista_desordenada.pop()
```

```
print(ultimo)
```

```
print(lista_desordenada)
```

```
# también elimina con el índice del campo
```

```
lista_desordenada.pop(i)
```

Ejercicio 15: Crear Lista de Listas

Crea una lista de listas donde cada sublista contiene tres números consecutivos y muestra la lista de listas.

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

```
print(lista_de_listas)
```

Ejercicio 16: Acceder a Elementos de Sublistas

Accede y muestra el primer elemento de la primera sublista.

```
print(lista_de_listas[0][0])
```

Ejercicio 17: Añadir Sublista

Añade una nueva sublista `[10, 11, 12]` usando `append()` y muestra la lista de listas.

```
lista_de_listas.append([10, 11, 12])
```

```
print(lista_de_listas)
```

Ejercicio 18: Fusionar dos Listas en Una Nueva Lista

Crea dos listas, fusiona sus elementos en una nueva lista usando el método `extend()` y muestra la nueva lista.

```
lista1 = [1, 2, 3]
```

```
lista2 = [4, 5, 6]
```

```
lista1.extend(lista2)
```

```
print(lista1)
```

Ejercicio 19: Lista de Strings

Crea una lista de strings con los nombres de cinco frutas y usa `sort()` para ordenarlos alfabéticamente.

```
frutas = ["manzana", "banana", "cereza", "dátil", "fresa"]
```

```
frutas.sort()
```

```
print(frutas)
```

Ejercicio 20: Eliminar Elemento Específico Usando pop()

Elimina el segundo elemento de la lista de frutas usando `pop()` y muestra la lista modificada y el elemento eliminado.

```
i = 2
```

```
elemento_eliminado = frutas.pop(i)
```

```
print(elemento_eliminado)
```

```
print(frutas)
```

Ejercicio 21: Eliminar todo el contenido de una lista clear()

```
lista.clear()
```

Ejercicio 22: Ejercicio práctico hacer un CRUD solo con las funciones append y del.

#con la posición del elemento en el vector actualizo y busco, con append agrego y con del elimino

```
print ("Bienvenido al Sistema de Inventario")
```

```
producto = []  
cantidad = []  
precio = []
```

```
i = True  
while i == True:
```

```
    x = int(input("Ingresa\n1--Crear Producto\n2--Buscar Producto\n3--Actualizar  
Producto\n4--Eliminar Producto\n5--Salir\nPara conocer el proceso: "))
```

```
    if x == 1:  
        apro = input("Ingresa nombre producto: ").capitalize()  
        acan = int(input("Ingresa cantidad del producto: "))  
        apre = float(input("Ingresa precio del producto: "))
```

```
        producto.append(apro)  
        cantidad.append(acan)  
        precio.append(apre)
```

```
        print("----- Producto almacenado correctamente :)-----")
```

```
    elif x == 2:  
        busPro = input("Ingresa producto a buscar: ").capitalize()  
        resultado = busPro in producto #trae un True o False
```

```
    if resultado == True:  
        print("---- Producto Encontrado :)-----")  
        elemento = producto.index(busPro)  
        #busca con el nombre del producto el índice para usarse así vector[i]  
  
        print("Nombre Producto: ", producto[elemento])  
        print("Cantidad del Producto: ", cantidad[elemento])  
        print("Precio del Producto: ", precio[elemento])
```

```

elif x == 3:
    busPro = input("Ingresa producto a Actualizar: ").capitalize()
    resultado = busPro in producto #trae un True o False

    if resultado == True:
        print("----Producto encontrado----")

        npro = input("Ingresa nombre nuevo producto: ").capitalize()
        ncan = int(input("Ingresa nueva cantidad del Producto: "))
        npre = float(input("Ingresa nuevo precio prodcuto: "))

        elemento = producto.index(busPro)
        #busca con el nombre del producto el índice para usarse así vector[i]
        producto[elemento] = npro
        cantidad[elemento] = ncan
        precio[elemento] = npre

        print("----Producto Actualizado Correctamente-----")

elif x == 4:
    busPro = input("Ingresa producto a Eliminar: ").capitalize()
    resultado = busPro in producto

    if resultado == True:
        print("----Producto encontrado----")
        elemento = producto.index(busPro)

        del producto[elemento]
        del cantidad[elemento]
        del precio[elemento]

        print("Producto eliminado Correctamente")

elif x == 5:
    print("Hasta Pronto")
    i = False

```



Al último ejercicio le vamos a meter un par de funciones mas para que quede completo:

Paso 1: Uso de `zip()`

El primer uso de `zip(producto, cantidad, precio)` crea un iterador de tuplas, donde cada tupla contiene elementos correspondientes de cada una de las tres listas. Esto significa que cada producto, su cantidad correspondiente y su precio se agrupan en una tupla.

Ejemplo:

Supongamos que tienes las siguientes listas:

```
producto = ['Manzana', 'Banana', 'Cereza']
```

```
cantidad = [10, 5, 7]
```

```
precio = [20.0, 15.0, 10.0]
```

Al aplicar `zip (producto, cantidad, precio)`, se genera:

```
[('Manzana', 10, 20.0), ('Banana', 5, 15.0), ('Cereza', 7, 10.0)]
```

Paso 2: Ordenación con `sorted()`

La función `sorted()` se usa para ordenar estas tuplas. Sin especificar ningún parámetro adicional, `sorted()` ordena las tuplas alfabéticamente por el primer elemento de cada tupla, es decir, el nombre del producto.

```
sorted([('Manzana', 10, 20.0), ('Banana', 5, 15.0), ('Cereza', 7, 10.0)])
```

Esto resultará en:

```
[('Banana', 5, 15.0), ('Cereza', 7, 10.0), ('Manzana', 10, 20.0)]
```

Paso 3: Desempaquetado con `zip(*iterable)`

Una vez que las tuplas están ordenadas, se utiliza `zip(*iterable)` para "desempaquetar" y "rezipar" las tuplas. El asterisco `*` en `*sorted(...)` desempaqueta la lista de tuplas de vuelta a varias listas de tuplas, que luego `zip()` reorganiza de nuevo en tres listas separadas.

Este paso convierte la lista de tuplas ordenadas de nuevo en tres listas, cada una correspondiendo a los productos, cantidades y precios, respectivamente, pero ahora en el nuevo orden alfabético.

Resultado:

Después de aplicar `zip(*sorted(...))` y asignarlo de vuelta a `producto, cantidad, precio`, tendrás:


```
producto = ['Banana', 'Cereza', 'Manzana']
```

```
cantidad = [5, 7, 10]
```

```
precio = [15.0, 10.0, 20.0]
```

paso 4: le agregamos también un imprimir todos los productos así

```
if producto:
    for p, c, pr in zip(producto, cantidad, precio):
        print(f"Producto: {p}, Cantidad: {c}, Precio: {pr}")
    else:
        print("No hay productos en el inventario.")
```

Uso de `zip()` en un ciclo para:

El método `zip()` toma como argumentos múltiples iterables, como listas, y los agrupa creando un iterador de tuplas. Cada tupla generada por `zip()` contiene un elemento de cada iterable que se pasa a `zip()`. Estos elementos están alineados por su índice, es decir, el primer elemento de cada iterable se agrupa en la primera tupla, el segundo en la segunda tupla, y así sucesivamente.

Este método de usar `zip()` con un bucle `for` y desempaqueado de tuplas es muy eficiente para trabajar con datos relacionados almacenados en múltiples listas, asegurando que los datos se mantengan sincronizados durante las operaciones.

Entonces `zip(producto, cantidad, precio)` crea un iterador de tuplas, donde cada tupla contiene un producto, su cantidad correspondiente y su precio, respectivamente.

Desempaquetado en el bucle `for`

El bucle `for` utiliza desempaquetado de tuplas para asignar los valores de cada tupla generada por `zip()` a las variables `p`, `c`, y `pr`. Así:

- `p` tomará el valor del elemento actual de la lista `producto`.
- `c` tomará el valor del elemento actual de la lista `cantidad`.
- `pr` tomará el valor del elemento actual de la lista `precio`.

Impresión de la Información

Dentro del bucle `for`, se usa la siguiente línea para imprimir los detalles de cada producto:

```
print(f"Producto: {p}, Cantidad: {c}, Precio: {pr}")
```

- Cada iteración del bucle imprimirá los detalles de un producto, mostrando su nombre, cantidad y precio en la salida estándar.

Ejemplo:

Si tenemos las siguientes listas:

```
producto = ['Manzana', 'Banana']
```

```
cantidad = [10, 5]
```

```
precio = [20.0, 15.0]
```

La salida del bucle `for` sería:

Producto: Manzana, Cantidad: 10, Precio: 20.0

Producto: Banana, Cantidad: 5, Precio: 15.0

Cada línea corresponde a una iteración del bucle, donde `p`, `c`, y `pr` toman los valores de las posiciones correspondientes de las listas `producto`, `cantidad`, y `precio`.

Ejercicio 23: Hacer un CRUD con `append`, `zip` y `pop`.

```
print("Bienvenido al Sistema de Inventario")

producto = []
cantidad = []
precio = []

i = True

while i:
    print("\nMenú de Opciones:")
    x = int(input("1--Crear Producto\n2--Buscar Producto\n3--Actualizar\n4--Eliminar Producto\n5--Listar Productos\n6--Ordenar Productos por\nNombre\n7--Invertir Orden de Productos\n8--Eliminar Todos los Productos\n9--\nSalir\nSelecciona una opción: "))

    if x == 1:
        apro = input("Ingresa nombre producto: ").capitalize()
        acan = int(input("Ingresa cantidad del producto: "))
```

```

apre = float(input("Ingresa precio del producto: "))

producto.append(apro)
cantidad.append(acan)
precio.append(apre)

print("----- Producto almacenado correctamente :-----")

elif x == 2:
    busPro = input("Ingresa producto a buscar: ").capitalize()
    if busPro in producto:
        print("---- Producto Encontrado :-----")
        elemento = producto.index(busPro)
        print("Nombre Producto: ", producto[elemento])
        print("Cantidad del Producto: ", cantidad[elemento])
        print("Precio del Producto: ", precio[elemento])
    else:
        print("Producto no encontrado.")

elif x == 3:
    busPro = input("Ingresa producto a Actualizar: ").capitalize()
    if busPro in producto:
        elemento = producto.index(busPro)
        print("----Producto encontrado----")
        npro = input("Ingresa nombre nuevo producto: ").capitalize()
        ncan = int(input("Ingresa nueva cantidad del Producto: "))
        npre = float(input("Ingresa nuevo precio producto: "))

        producto[elemento] = npro
        cantidad[elemento] = ncan
        precio[elemento] = npre

        print("----Producto Actualizado Correctamente-----")
    else:
        print("Producto no encontrado.")

elif x == 4:
    busPro = input("Ingresa producto a Eliminar: ").capitalize()
    if busPro in producto:
        print("----Producto encontrado y se procede a eliminar----")
        elemento = producto.index(busPro)

```

```
    producto.pop(elemento) # Uso de pop para eliminar
    cantidad.pop(elemento)
    precio.pop(elemento)
    print("Producto eliminado Correctamente")
else:
    print("Producto no encontrado.")
```

```
elif x == 5:
    if producto:
        for p, c, pr in zip(producto, cantidad, precio):
            print (f"Producto: {p}, Cantidad: {c}, Precio: {pr}")
    else:
        print ("No hay productos en el inventario.")
```

```
elif x == 6:
    #Ordena los productos alfabéticamente con zip y mantiene la relación
    entre los productos, sus cantidades y precios, sorted ordena alfabéticamente con
    la primera tupla.
    producto, cantidad, precio=zip(*sorted(zip(producto, cantidad, precio)))
    print("Productos ordenados alfabéticamente.")
```

```
elif x == 7:
    #Ordena los productos al revés
    producto.reverse()
    cantidad.reverse()
    precio.reverse()
    print("Orden de productos invertido.")
```

```
elif x == 8:
    #elimina los valores de las tuplas
    producto.clear()
    cantidad.clear()
    precio.clear()
    print("Todos los productos han sido eliminados.")
```

```
elif x == 9:
    print("Hasta Pronto")
    i = False
```