

## Laboratorio #3 Programación II

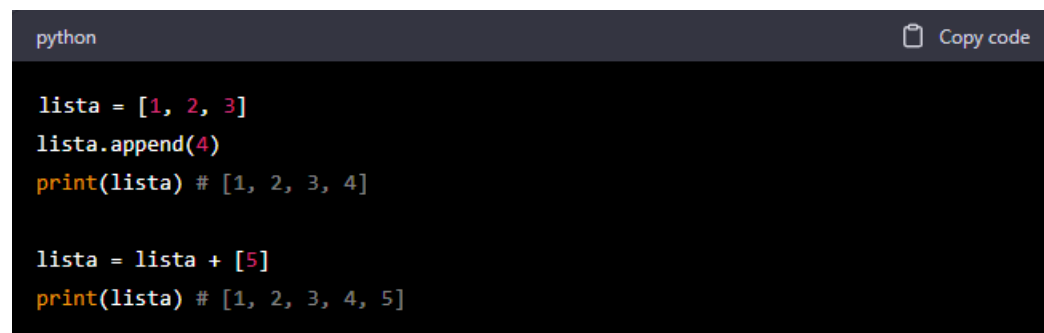
### Warren Bonilla Jiménez

#### 1- De qué forma podemos agregar, eliminar, modificar y recorrer una estructura tipo list?

En Python, una estructura de datos de tipo lista es una secuencia ordenada de elementos que se pueden agregar, eliminar, modificar y recorrer de varias maneras.

#### Agregar elementos a una lista:

Para agregar elementos a una lista, se puede utilizar el método `append()` o el operador de suma `+`.

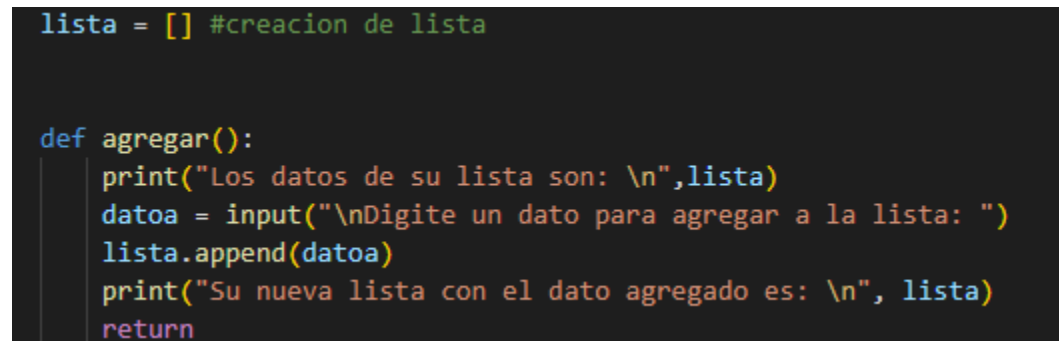


```
python Copy code

lista = [1, 2, 3]
lista.append(4)
print(lista) # [1, 2, 3, 4]

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

Otra forma para agregar datos dentro de la lista es de la siguiente manera como se encuentra en la imagen



```
lista = [] #creacion de lista

def agregar():
    print("Los datos de su lista son: \n", lista)
    datoa = input("\nDigite un dato para agregar a la lista: ")
    lista.append(datoa)
    print("Su nueva lista con el dato agregado es: \n", lista)
    return
```

## Eliminar elementos de una lista:

Para eliminar elementos de una lista, se puede utilizar el método `remove()` o la palabra clave `del`.

```
python Copy code  
  
lista = [1, 2, 3, 4, 5]  
lista.remove(3)  
print(lista) # [1, 2, 4, 5]  
  
del lista[2]  
print(lista) # [1, 2, 5]
```

Otra forma para eliminar datos dentro de la lista es de la siguiente manera como se encuentra en la imagen

```
lista = [] #creacion de lista
```

```
def eliminar():  
    print("Los datos de su lista son: \n", lista)  
    dato = input("\nDigite un dato para eliminarlo de la lista: ")  
    lista.remove(dato)  
    print("Su nueva lista con el dato eliminado es: \n", lista)  
    return
```

## Modificar elementos de una lista:

Para modificar elementos de una lista, se puede acceder a ellos mediante su índice y asignarles un nuevo valor.

```
python Copy code  
  
lista = [1, 2, 3]  
lista[1] = 4  
print(lista) # [1, 4, 3]
```

Otra forma para eliminar datos dentro de la lista es de la siguiente manera como se encuentra en la imagen

```
lista = [] #creacion de lista
```

```
def modificar():
    print("Los datos de su lista son: \n",lista)
    datom = input("\nDigite el dato de los anteriores que le gustaria modificar de la lista: ")
    posi = lista.index(datom) #se obtiene la posicion del dato a modificar
    datom2 = input("\nDigite el nuevo dato a modificar: ") # se obtiene el nuevo dato a modificar
    lista[posi] = datom2
    print("Su nueva lista modificada es: \n",lista)
    return
```

## Recorrer una lista:

Para recorrer los elementos de una lista, se puede utilizar un bucle for o un bucle while.

```
python Copy code

lista = [1, 2, 3, 4, 5]
for elemento in lista:
    print(elemento)

# Output:
# 1
# 2
# 3
# 4
# 5

i = 0
while i < len(lista):
    print(lista[i])
    i += 1

# Output:
# 1
# 2
# 3
# 4
# 5
```

## 2- De que forma podemos contar la cantidad de elementos que se encuentran en una colección tipo list

La función len() devuelve el número de elementos en la lista.

```
python Copy code

mi_lista = [1, 2, 3, 4, 5]
cantidad_elementos = len(mi_lista)
print(cantidad_elementos) # Output: 5
```

En el ejemplo anterior, la función len() devuelve el número de elementos de la lista mi lista, que es 5. Luego, asignamos este valor a la variable cantidad de elementos y la imprimimos en la consola.

### 3- Que entiende por indexación de los elementos en una colección de datos.

La indexación de los elementos en una colección de datos se refiere a la capacidad de acceder a los elementos individuales de la colección mediante un índice numérico. En Python, las colecciones de datos que admiten la indexación incluyen las listas, tuplas, cadenas y otros tipos de colecciones.

En una colección indexada, cada elemento se almacena en una posición única, conocida como índice, que se utiliza para acceder a él. Los índices en Python comienzan en 0, lo que significa que el primer elemento de una colección tiene un índice de 0, el segundo elemento tiene un índice de 1, y así sucesivamente. Por ejemplo:

```
python Copy code  
  
mi_lista = ['a', 'b', 'c', 'd', 'e']  
primer_elemento = mi_lista[0] # 'a'  
segundo_elemento = mi_lista[1] # 'b'  
tercer_elemento = mi_lista[2] # 'c'
```

En el ejemplo anterior, `mi_lista` es una lista de caracteres y se utilizan los índices para acceder a los primeros tres elementos de la lista. El primer elemento de la lista se encuentra en la posición 0, el segundo en la posición 1 y el tercer elemento en la posición 2.

Es importante tener en cuenta que si intentamos acceder a un índice que está fuera de los límites de la colección, se producirá un error de índice fuera de rango. Por ejemplo:

```
python Copy code  
  
cuarto_elemento = mi_lista[4] # IndexError: list index out of range
```

En este ejemplo, se intenta acceder al quinto elemento de la lista (índice 4), que no existe, lo que produce un error.

En resumen, la indexación de los elementos en una colección de datos es una técnica común para acceder y manipular elementos individuales de una colección utilizando un índice numérico.