Universidad Internacional de las Américas
Escuela de Ingeniería Informática

Laboratorio #3

Estudiante

Jason Brenes Salgado

Profesor

Carlos González Romero

2023

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

Las listas son secuencias mutables, es decir, sus elementos pueden ser modificados (se pueden añadir nuevos ítems, actualizar o eliminar).

Para **añadir** un nuevo elemento a una lista se utiliza el método `append()` y para añadir varios elementos, el método `extend()`

`Ejemplo:`

```
>>> vocales = ['a']
>>> vocales.append('e')  # Añade un elemento
>>> vocales
['a', 'e']

>>> vocales.extend(['i', 'o', 'u'])  # Añade un grupo de elementos
>>> vocales
['a', 'e', 'i', 'o', 'u']
```

Es posible **modificar** un elemento de una lista en Python con el operador de asignación `=`.Para ello, lo único que necesitas conocer es el índice del elemento que quieres modificar o el rango de índices:

Ejemplo:

```
>>> vocales = ['o', 'o', 'o', 'o', 'u']

# Actualiza el elemento del índice 0
>>> vocales[0] = 'a'
>>> vocales
['a', 'o', 'o', 'o', 'u']

# Actualiza los elementos entre las posiciones 1 y 2
>>> vocales[1:3] = ['e', 'i']
>>> vocales
['a', 'e', 'i', 'o', 'u']
```

En Python se puede **eliminar** un elemento de una lista de varias formas.
Con la sentencia `del` se puede eliminar un elemento a partir de su índice:

Ejemplo:

```
# Elimina el elemento del índice 1
>>> vocales = ['a', 'e', 'i', 'o', 'u']
>>> del vocales[1]
>>> vocales
['a', 'i', 'o', 'u']

# Elimina los elementos con índices 2 y 3
>>> vocales = ['a', 'e', 'i', 'o', 'u']
>>> del vocales[2:4]
>>> vocales
['a', 'e', 'u']

# Elimina todos los elementos
>>> del vocales[:]
>>> vocales
[]
```

Además de la sentencia **`del`**, podemos usar los métodos **`remove()`** y **`pop([i])`.** **`remove()`** elimina la primera ocurrencia que se encuentre del elemento en una lista. Por su parte, **`pop([i])`** obtiene el elemento cuyo índice sea igual a `i` y lo elimina de la lista. Si no se especifica ningún índice, recupera y elimina el último elemento.

Ejemplo

```
>>> letras = ['a', 'b', 'k', 'a', 'v']

# Elimina la primera ocurrencia del carácter a
>>> letras.remove('a')
>>> letras
['b', 'k', 'a', 'v']

# Obtiene y elimina el último elemento
>>> letras.pop()
'v'
>>> letras
['b', 'k', 'a']
```

Finalmente, es posible eliminar todos los elementos de una lista a través del método clear():

Ejemplo:

```
>>> letras = ['a', 'b', 'c']
>>> letras.clear()
>>> letras
[]
```

Para recorrer una lista de Python se puede realizar usando un **bucle for :**

Ejemplo:

```
mylist = [1,4,7,3,21]



for x in mylist:

  print(x)
```

2. De que forma podemos contar la cantidad de elementos que se encuentran en una colección tipo <list>

Las listas en Python pueden almacenar múltiples elementos de diferentes tipos de datos.

La función incorporada **len()** en Python devuelve el número total de elementos en una lista, sin tener en cuenta el tipo de elementos que contiene.

Ejemplo:

```
list1 = ["God","Belief",10,31,"Human"]



print("The total number of elements in the list: ", len(list1))
```

Otra forma básica de contar el número de elementos es hacer uso del bucle **for**. El bucle comienza con la cuenta establecida en 0 y continúa hasta el último elemento; el recuento se incrementa en uno cada vez que se encuentra un elemento de la lista en la iteración del bucle.

El siguiente código utiliza el bucle **for** para obtener el número de elementos de la lista.

Ejemplo:

```
list2 = ["Hey",20,14,"Look","An Example List"]



def total_elements(list):

    count = 0

    for element in list:

        count += 1

    return count



print("The total number of elements in the list: ", total_elements(list2))
```

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

A mi parecer entiendo que es agregar nuevos datos en una lista determinada , la cual se va actualizando con el tiempo