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

- *Agregar:*

Se utiliza el método `append()`y para varios elementos el método `extend()`.

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

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

Podemos utilizar el operador `+` para unir dos listas en una sola.

```
lista_1 = [1, 2, 3]
lista_2 = [4, 5, 6]
nueva_lista = lista_1 + lista_2
print(nueva_lista)
```

Podemos utilizar el operador `*` repite el contenido de una lista `n` veces.

```
numeros = [1, 2, 3]
numeros *= 3
print(numeros)
```

Es posible añadir un elemento en una posición concreta de una lista con el método `insert(índice, elemento)`. Los elementos cuyo índice sea mayor a `índice` se desplazan una posición a la derecha:

```
vocales = ['a', 'e', 'u']
vocales.insert(2, 'i')
print(vocales)
```

- *Eliminar:*

Con la sentencia `del` se puede eliminar un elemento a partir de su índice

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

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

# Elimina todos los elementos
del vocales[:]
print(vocales)
```

Los métodos `remove()` y `pop([i])`:

`remove()`: elimina la primera ocurrencia que se encuentre del elemento en una lista.

`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.

```
letras = ['a', 'b', 'k', 'a', 'v']
# Elimina la primera ocurrencia del carácter a
letras.remove('a')
print(letras)
['b', 'k', 'a', 'v']
# Obtiene y elimina el último elemento
letras.pop()
'v'
print(letras)
```

Es posible eliminar todos los elementos de una lista a través del método `clear():`

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

- *Modificar:*

Es posible con el operador de asignación `=.` Para ello, lo único que necesitas conocer es el índice del elemento que quieres modificar:

```
vocales = ['o', 'o', 'o', 'o', 'u']
# Actualiza el elemento del índice 0
vocales[0] = 'a'
print(vocales)

# Actualiza los elementos entre las posiciones 1 y 2
vocales[1:3] = ['e', 'i']
print(vocales)
```

- *Recorrer:*

Para recorrer una lista, utilizaremos la siguiente estructura:

```
colores = ['azul', 'blanco', 'negro']
for color in colores:
        print(color)
```

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

La forma predeterminada de obtener la longitud de un objeto es usar la función integrada `len()`. Devuelve un valor entero que indica el número total de elementos en el objeto.
secuencia (como una lista, string o rango), o colección (como un diccionario o conjunto)

```
if __name__ == '__main__':
    nums = [1, 2, 3, 4, 5] # a list
    print(len(nums)) # imprimir 5
```

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

La indexación es la extracción de datos por número de posición, de la cual empieza a contar desde 0. Por ejemplo; quiero sacar la letra “m” de la siguiente palabra: Programar, lo que debemos hacer es encerrar el número de posición de ese carácter entre corchetes y el programa devolverá ese carácter.

```
print("Programar"[6])
```