Sesión 4. Listas y repeticiones

Ejemplos

1. Una lista puede ser heterogénea, es decir, tener elementos de diferentes tipos. Por ejemplo

```
lista = [34, 'un texto', True, -23.5, 'correo@gmail.com']
```

Los elementos de una lista están indexados con los números $0, 1, 2, \ldots, n-1$, donde n es la longitud o tamaño de la lista. Podemos acceder a un elemento de la lista usando el índice que le corresponde:

```
elemento = lista[1]
print(elemento)  # Imprime 'un texto'
elemento = lista[0]
print(elemento)  # Imprime 34
```

También es posible acceder a los elementos usando índices negativos $-1, -2, \ldots$, donde el elemento -1 es el último elemento (de índice $n-1$), el elemento -2 es el penúltimo (índice $n-2$), etcétera.

```
elemento = lista[-1]
print(elemento)  # Imprime 'correo@gmail.com'
elemento = lista[-3]
print(elemento)  # Imprime True
```

2. Podemos comprobar si un elemento x pertenece a una lista usando la expresión «`x in lista`» como en el siguiente ejemplo

```
lista = [1, 23, 'python', 2.5]
print(5 in lista)  # Imprime False porque 5 no está en lista
print(23 in lista) # Imprime True porque 23 sí está en lista
```

Si un elemento x está en una lista, podemos usar el método `lista.index(x)` para obtener el índice del elemento. Por ejemplo

```
if x in lista:               # Índice de la primera aparición
    indice = lista.index(x)  # de x en lista
```

3. Una cadena es muy parecida a una lista de caracteres, de hecho las listas y las cadenas comparten varias operaciones y métodos. Por ejemplo, los caracteres de una cadena están indexados igual que una lista, con enteros $0, 1, \ldots$ y también es posible seleccionar subcadenas con expresiones de la forma `cadena[a:b]`

```
cadena = 'México'
cadena[1]    # Cadena 'é'
cadena[-2]   # Cadena 'c'
cadena[1:4]  # Cadena 'éxi'
```

Ejercicios

1. Haz un programa que acepte un entero n en la terminal y determine si es un número primo (que sus únicos divisores positivos son 1 y n) o es un número compuesto.

2. En el siguiente código `lista3` debe contener la *intersección* de `lista1` y `lista2`, es decir, aquellos elementos que aparecen en ambas listas. Completa el código para obtener la intersección de `lista1` y `lista2` sin importar los elementos que tengan.

```
lista1 = [1, 43, 101, 2938, -23, -34]
lista2 = [3, 1, 52, 50, 0, 101]
# Faltan varias líneas de código
print(lista3) # Imprime [1, 101]
```

   Es posible que `lista3` tenga elementos repetidos, ¿qué harías para evitar estas repeticiones?

3. Se puede definir una *distancia* entre cadenas de igual longitud: Sean $a = a_1 a_2 \cdots a_n$ y $b = b_1 b_2 \cdots b_n$ dos cadenas de longitud n, se define la distancia

$$d(a,b) \stackrel{\text{def}}{=} |\{1 \leq i \leq n \mid a_i \neq b_i\}|,$$

   es decir, $d(a,b)$ es el número de posiciones donde son distintos los caracteres de a y b.

   Haz un programa que pida dos cadenas en la terminal y que imprima su distancia. Si las cadenas no tienen la misma longitud se debe mostrar un mensaje de error.