ej_2

September 29, 2025

0.1 Estructuras de Datos

[1]:
```
# 1. Crear una lista que contenga nombres de ciudades del mundo que contenga␣
 ↪más de 5 elementos e imprimir por pantalla
ciudades = ["Buenos Aires", "Madrid", "París", "Tokio", "Nueva York", "Londres"]
print(ciudades)
```

```
['Buenos Aires', 'Madrid', 'París', 'Tokio', 'Nueva York', 'Londres']
```

[2]:
```
# 2. Imprimir por pantalla el segundo elemento de la lista
print(ciudades[1])
```

```
Madrid
```

[3]:
```
# 3. Imprimir por pantalla del segundo al cuarto elemento
print(ciudades[1:4])
```

```
['Madrid', 'París', 'Tokio']
```

[4]:
```
# 4. Visualizar el tipo de dato de la lista
print(type(ciudades))
```

```
<class 'list'>
```

[5]:
```
# 5. Visualizar todos los elementos de la lista a partir del tercero de manera␣
 ↪genérica, es decir, sin explicitar la posición del último elemento
print(ciudades[2:])
```

```
['París', 'Tokio', 'Nueva York', 'Londres']
```

[6]:
```
# 6. Visualizar los primeros 4 elementos de la lista
print(ciudades[:4])
```

```
['Buenos Aires', 'Madrid', 'París', 'Tokio']
```

[7]:
```
# 7. Agregar una ciudad más a la lista que ya exista y otra que no ¿Arroja␣
 ↪algún tipo de error?
ciudades.append("Madrid") # Agrega una ciudad que ya existe
ciudades.append("Roma")   # Agrega una ciudad que no existe
print(ciudades)
# No arroja error al agregar elementos duplicados a una lista.
```

```
['Buenos Aires', 'Madrid', 'París', 'Tokio', 'Nueva York', 'Londres', 'Madrid',
'Roma']
```

[8]:
```
# 8. Agregar otra ciudad, pero en la cuarta posición
ciudades.insert(3, "Sídney")
print(ciudades)
```

```
['Buenos Aires', 'Madrid', 'París', 'Sídney', 'Tokio', 'Nueva York', 'Londres',
'Madrid', 'Roma']
```

[9]:
```
# 9. Concatenar otra lista a la ya creada
otras_ciudades = ["Berlín", "Toronto"]
ciudades.extend(otras_ciudades)
print(ciudades)
```

```
['Buenos Aires', 'Madrid', 'París', 'Sídney', 'Tokio', 'Nueva York', 'Londres',
'Madrid', 'Roma', 'Berlín', 'Toronto']
```

[10]:
```
# 10. Encontrar el índice de la ciudad que en el punto 7 agregamos duplicada.␣
 ↪¿Se nota alguna particularidad?
indice_madrid = ciudades.index("Madrid")
print(f"El índice de la primera aparición de 'Madrid' es: {indice_madrid}")
# La particularidad es que .index() solo devuelve el índice de la primera␣
 ↪aparición del elemento.
```

```
El índice de la primera aparición de 'Madrid' es: 1
```

[11]:
```
# 11. ¿Qué pasa si se busca un elemento que no existe?
try:
    ciudades.index("Bogotá")
except ValueError as e:
    print(f"Ocurrió un error: {e}")
# Si se busca un elemento que no existe, se produce un error de tipo ValueError.
```

```
Ocurrió un error: 'Bogotá' is not in list
```

[12]:
```
# 12. Eliminar un elemento de la lista
ciudades.remove("Londres")
print(ciudades)
```

```
['Buenos Aires', 'Madrid', 'París', 'Sídney', 'Tokio', 'Nueva York', 'Madrid',
'Roma', 'Berlín', 'Toronto']
```

[13]:
```
# 13. ¿Qué pasa si el elemento a eliminar no existe?
try:
    ciudades.remove("Miami")
except ValueError as e:
    print(f"Ocurrió un error: {e}")
# Si el elemento a eliminar no existe, se produce un error de tipo ValueError.
```

```
Ocurrió un error: list.remove(x): x not in list
```

[14]:
```
# 14. Extraer el último elemento de la lista, guardarlo en una variable e␣
↪imprimirlo
ultima_ciudad = ciudades.pop()
print(f"El último elemento extraído es: {ultima_ciudad}")
print(f"La lista después de extraer el último elemento es: {ciudades}")
```

```
El último elemento extraído es: Toronto
La lista después de extraer el último elemento es: ['Buenos Aires', 'Madrid',
'París', 'Sídney', 'Tokio', 'Nueva York', 'Madrid', 'Roma', 'Berlín']
```

[15]:
```
# 15. Mostrar la lista multiplicada por 4
print(ciudades * 4)
```

```
['Buenos Aires', 'Madrid', 'París', 'Sídney', 'Tokio', 'Nueva York', 'Madrid',
'Roma', 'Berlín', 'Buenos Aires', 'Madrid', 'París', 'Sídney', 'Tokio', 'Nueva
York', 'Madrid', 'Roma', 'Berlín', 'Buenos Aires', 'Madrid', 'París', 'Sídney',
'Tokio', 'Nueva York', 'Madrid', 'Roma', 'Berlín', 'Buenos Aires', 'Madrid',
'París', 'Sídney', 'Tokio', 'Nueva York', 'Madrid', 'Roma', 'Berlín']
```

[16]:
```
# 16. Crear una tupla que contenga los números enteros del 1 al 20
numeros = tuple(range(1, 21))
print(numeros)
```

```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
```

[17]:
```
# 17. Imprimir desde el índice 10 al 15 de la tupla
print(numeros[10:16])
```

```
(11, 12, 13, 14, 15, 16)
```

[18]:
```
# 18. Evaluar si los números 20 y 30 están dentro de la tupla
print(f"¿El número 20 está en la tupla? {20 in numeros}")
print(f"¿El número 30 está en la tupla? {30 in numeros}")
```

```
¿El número 20 está en la tupla? True
¿El número 30 está en la tupla? False
```

[19]:
```
# 19. Con la lista creada en el punto 1, validar la existencia del elemento␣
↪'París' y si no existe, agregarlo. Utilizar una variable e informar lo␣
↪sucedido.
informe = ""
if 'París' not in ciudades:
    ciudades.append('París')
    informe = "La ciudad 'París' no existía y ha sido agregada."
else:
    informe = "La ciudad 'París' ya existía en la lista."

print(informe)
print(ciudades)
```

```
La ciudad 'París' ya existía en la lista.
['Buenos Aires', 'Madrid', 'París', 'Sídney', 'Tokio', 'Nueva York', 'Madrid',
'Roma', 'Berlín']
```

[21]:
```python
# 20. Mostrar la cantidad de veces que se encuentra un elemento específico␣
 ↪dentro de la tupla y de la lista
elemento_buscar_lista = "Madrid"
elemento_buscar_tupla = 15

cantidad_lista = ciudades.count(elemento_buscar_lista)
cantidad_tupla = numeros.count(elemento_buscar_tupla)

print(f"'{elemento_buscar_lista}' aparece {cantidad_lista} veces en la lista.")
print(f"{elemento_buscar_tupla} aparece {cantidad_tupla} veces en la tupla.")
```

```
'Madrid' aparece 2 veces en la lista.
15 aparece 1 veces en la tupla.
```

[22]:
```python
# 21. Convertir la tupla en una lista
lista_desde_tupla = list(numeros)
print(lista_desde_tupla)
print(type(lista_desde_tupla))
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
<class 'list'>
```

[23]:
```python
# 22. Desempaquetar solo los primeros 3 elementos de la tupla en 3 variables
a, b, c, *resto = numeros
print(f"Primer elemento: {a}")
print(f"Segundo elemento: {b}")
print(f"Tercer elemento: {c}")
```

```
Primer elemento: 1
Segundo elemento: 2
Tercer elemento: 3
```

[25]:
```python
# 23. Crear un diccionario utilizando la lista crada en el punto 1, asignandole␣
 ↪la clave "ciudad". Agregar tambien otras claves, como puede ser "Pais" y␣
 ↪"Continente".
# Este punto asume que se quiere crear un diccionario donde cada elemento de la␣
 ↪lista de ciudades es parte de un valor asociado a la clave "ciudad".
# Aquí crearemos un diccionario con una clave 'ciudades' que contenga la lista,␣
 ↪y claves adicionales para país y continente de forma genérica o como listas␣
 ↪separadas si aplica.
# Optaremos por un diccionario donde cada ciudad es una clave y su valor es␣
 ↪otro diccionario con País y Continente, aunque la lista original solo tiene␣
 ↪ciudades.
# Para simplificar, usaremos un ejemplo con datos inventados para algunas␣
 ↪ciudades.
```

```
diccionario_ciudades = {
    "Buenos Aires": {"Pais": "Argentina", "Continente": "América"},
    "Madrid": {"Pais": "España", "Continente": "Europa"},
    "París": {"Pais": "Francia", "Continente": "Europa"},
    "Tokio": {"Pais": "Japón", "Continente": "Asia"},
    "Nueva York": {"Pais": "USA", "Continente": "América"},
    "Roma": {"Pais": "Italia", "Continente": "Europa"},
    "Sídney": {"Pais": "Australia", "Continente": "Oceanía"},
    "Berlín": {"Pais": "Alemania", "Continente": "Europa"},
    "Toronto": {"Pais": "Canadá", "Continente": "América"}
}
print(diccionario_ciudades)
```

```
{'Buenos Aires': {'Pais': 'Argentina', 'Continente': 'América'}, 'Madrid':
{'Pais': 'España', 'Continente': 'Europa'}, 'París': {'Pais': 'Francia',
'Continente': 'Europa'}, 'Tokio': {'Pais': 'Japón', 'Continente': 'Asia'},
'Nueva York': {'Pais': 'USA', 'Continente': 'América'}, 'Roma': {'Pais':
'Italia', 'Continente': 'Europa'}, 'Sídney': {'Pais': 'Australia', 'Continente':
'Oceanía'}, 'Berlín': {'Pais': 'Alemania', 'Continente': 'Europa'}, 'Toronto':
{'Pais': 'Canadá', 'Continente': 'América'}}
```

[26]:
```
# 24. Imprimir las claves del diccionario
print(diccionario_ciudades.keys())
```

```
dict_keys(['Buenos Aires', 'Madrid', 'París', 'Tokio', 'Nueva York', 'Roma',
'Sídney', 'Berlín', 'Toronto'])
```

[27]:
```
# 25. Imprimir las ciudades a través de su clave
# Asumiendo que "ciudades" se refiere a las claves del diccionario que son los␣
 ↪nombres de las ciudades
print(diccionario_ciudades.keys())

# Si la intención era imprimir los valores asociados a la clave "ciudad" en un␣
 ↪diccionario diferente (como se planteó en el punto 23),
# y si tuviéramos un diccionario con la estructura {"ciudad":␣
 ↪[lista_de_ciudades], ...}, el código sería:
# print(mi_otro_diccionario["ciudad"])
```

```
dict_keys(['Buenos Aires', 'Madrid', 'París', 'Tokio', 'Nueva York', 'Roma',
'Sídney', 'Berlín', 'Toronto'])
```