Checkpoint nº 4: teoría

1. **¿Cuál es la diferencia entre una lista y una tupla en Python?**

Ambas son estructuras de datos básicas de Python (built-in) y se representan como una serie de objetos de Python, encerrados entre paréntesis, las tuplas, y corchetes, las listas.

```
lista_ejemplo = [1, 2, 3, 4, 5,]  #Lista
tupla_ejemplo = (1, 2, 3, 4, 5,)  #Tupla
```

Los objetos que podemos almacenar en ambas pueden ser cualquier tipo de datos, incluso podemos anidar datos complejos, como otras listas, tuplas o diccionarios. La forma de acceder a los datos individuales de ambas es igual:

```
lista_ejemplo[3]  #El valor obtenido es 4
tupla_ejemplo[3]  #El valor obtenido es 4
```

La diferencia fundamental es que las listas son MUTABLES, es decir, están diseñadas para almacenar datos que pueden variar, mientras que las tuplas son INMUTABLES, es decir, están diseñadas para almacenar datos que no varian (o varían muy poco) Debido a eso, la clase 'list' (lista) tiene muchos métodos que nos permiten modificarlas: lst.append(), lst.extend(), lst.insert(), lst.remove(), lst.pop, lst.clear(), lst.sort(), lst.reverse(), lst.index(),...

La clase 'tuple' (tupla) no tiene métodos específicos para modificarlas, aunque sí se pueden hacer cambios de una forma indirecta. Por ejemplo, para añadir un elemento:

```
tupla += (elemento,)  #Hemos añadido el valor 'elemento' al final de la tupla
```

Como conclusión, las tuplas deben ser usadas para almacenar datos fijos, que no varían en toda la aplicación, mientras que si queremos manejar datos agrupados que pueden cambiar durante la ejecución de una aplicación, debemos usar listas.

2. **¿Cuál es el orden de las operaciones?**

Dada una operación múltiple con los operadores matemáticos, el orden en el que se deben realizar las operaciones es:

1. Paréntesis
2. Exponencial
3. Multiplicación
4. División
5. Suma
6. Resta

```
5 + 4 - (2 + 3) * 6 / 2 + 2**3
5 + 4 - 5 * 6 / 2 + 2**3
5 + 4 - 5 * 6 / 2 + 8
5 + 4 - 30 / 2 + 8
5 + 4 - 15 + 8
9 - 15 + 8
- 6 + 8
2
```

3. **¿Qué es un diccionario Python?**

Es una estructura de datos básica (built-in) que nos permite almacenar multiples datos del tipo clave-valor. Veámoslo con un ejemplo:

```
diccionario_ejemplo = {'incidencia': 'Ordenador no arranca', 'fecha': '25-02-
2025', 'tecnico asignado': 'Iñaki Mendigutxia', 'resuelto': True, 'tiempo
resolución(horas)'; 2.5 }
```

Las claves para este caso son: 'incidencia', 'fecha', 'técnico asignado'', 'resuelto', 'tiempo resolución(horas)' y los valores asociados son los valores que se encuentran detrás de los 2 puntos. Definida de esta manera, la recuperación de información es muy sencilla y trasparente:

```
diccionario_ejemplo['tecnico asignado']  #El valor que entrega es 'Iñaki
Mendigutxia'
```

También, como las listas, es una estructura mutable y tiene definidos muchos métodos para su manipulación: dct.get(), dct.keys(), dct,values(), dct.pop(), dct.items(),...

4. **¿Cuál es la diferencia entre el método ordenado y la función de ordenación?**

El *método* sort(*key*, *reverse*) se aplica a un objeto lista y produce la ordenación de la **misma lista** sobre la que se aplica, de acuerdo a un criterio definido por el parámetro *key* y *reverse*. Esta ordenación se realiza en el mismo objeto, no creándose otro objeto lista **diferente**. De hecho, esta función entrega el valor *None*, salvo error.

```
lista = [1, 2, 3]
print(lista.sort())  #Salida: None
print(lista)         #Salida: [3, 2, 1]
```

La *función* sorted(*iterable*, *key*, *reverse*), aplicada a una lista, crea una **nueva** lista, diferente de la original, ordenada por el criterio *key* y *reverse*. Hay que señalar que esta función se puede aplicar sobre **cualquier** iterable, no solo sobre listas.

```
lista = [1, 2, 3]
lista2 = sorted(lista)
print(lista)   #Salida: [1, 2, 3]
print(lista2)  #Salida: [3, 2, 1]
```

5. **¿Qué es un operador de reasignación?**

Los operadores de reasignación son operadores matemáticos que asignan a un mismo objeto existente el valor que tenía inicialmente ese objeto, modificado según la operación definida y otro valor que aparece en la expresión. Son los siguientes: =, +=, -=, *=, /=, **=, //=, %=

```
a = 100  #Estos valores se usan para cada operación de forma individual
b = 20   #Estos valores se usan para cada operación de forma individual

a += b   #Resultado a = 120, equivalente  a = a + b
a -= b   #Resultado a = 80, equivalente a = a - b
a *= b   #Resultado a = 2000, equivalente a = a * b
a /= b   #Resultado a = 50, equivalente a = a / b
a **= b  #Resultado a = 1 e40, equivalente a = a ** b
a //= b  #Resultado a = 5, equivalente a = a // b
a %= b   #Resultado a = 0, equivalente a = a % b
```