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

La principal diferencia es que una tupla es inmutable y una lista es mutable.
Esto quiere decir que una tupla no puede ser modificada, por lo que no se pueden añadir, borrar ni reemplazar elementos. Para poder hacer cambios, habría que hacerlo dentro de nuevas variables.

Ejemplo de eliminación de un elemento de una tupla, PERO dentro de una nueva variable que tiene el mismo nombre que la anterior:

Tupla original:

```
ingredientes = ('cebolla','zanahoria', 'tomate', 'carne picada')
print(ingredientes)
```

Si la ejecutamos nos da:

```
('cebolla', 'zanahoria', 'tomate', 'carne picada')
```

Creamos una nueva tupla cuya variable también se llame “ingredientes” y en la que eliminaremos el último elemento “carne picada”:

```
ingredientes = ingredientes[:-1]
print(ingredientes)
```

Resultado:

```
('cebolla', 'zanahoria', 'tomate')
```

Sin embargo, en una lista puedes hacer las modificaciones que quieras.
Por ejemplo, eliminar un elemento de la lista:

```
ingredientes = ['cebolla','zanahoria', 'tomate', 'carne picada']
print(ingredientes)

ingredientes.remove('cebolla')
print(ingredientes)
```

Y te devuelve:

```
['zanahoria', 'tomate', 'carne picada']
```

En función del tipo de datos que manejes, preferirás utilizar tuplas para que dichos datos no puedan ser cambiados, o listas.

La sintaxis utilizada en cada una de ellas también es diferente. En las tuplas utilizamos paréntesis, mientras que en las listas usamos corchetes. Ejemplo:

Tupla:

```
ingredientes = ('cebolla','zanahoria',  'tomate', 'carne picada')
print(ingredientes)
```

Lista:

```
ingredientes = ['cebolla','zanahoria',  'tomate', 'carne picada']
print(ingredientes)
```

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

Python respeta el orden aritmético. El orden a la hora de operar es el siguiente:

Parans (paréntesis)
Exponents (exponentes)
Multiplication (multiplicación)
Division (división)
Addition (suma)
Subtraction (resta)

Aunque la multiplicación y la división pueden invertir el orden entre ambas, ya que obtendríamos el mismo resultado. Debe de haber alguna variación más en cuanto al orden, pero en todas ellas el resultado es el mismo. Y al final, eso es lo que importa.

3.- ¿Qué es un diccionario Python?

Podemos pensar en un diccionario Python como en un diccionario de toda la vida, ya que ambos almacenan datos.

Un diccionario Python cuenta con claves y valores. Ejemplo:

```
alumnos_guarderia = {
    "1": "Mateo",
    "2": "Aida",
    "3": "Ian",
    "4": "Emma"
}

print(alumnos_guarderia)
```

En este ejemplo, las claves son 1, 2, 3 y 4, mientras que los valores son los nombres de los alumnos (Mateo, Aida, Ian y Emma). Todos ellos se encuentran dentro de una variable (alumnos_guarderia).

A nivel de sintaxis, un diccionario utiliza llaves {} para almacenar los datos.
Algo más a tener en cuenta es que los diccionarios son mutables, por lo que podemos modificarlos.

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

A diferencia del método ordenado[.sort()], la función sorted permite almacenar el valor de otra variable. De este modo, la lista original no se queda como estaba en un primer momento y se puede utilizar en apartados en los que no quieras un listado ordenado.

Por lo demás, ambas ordenan los datos alfanuméricos de la A a la Z, o viceversa y los números de menor a mayor valor y viceversa.

Ejemplos:

Método ordenado:

```
# Ejemplo con datos alfanuméricos:
compra = ['yogures', 'manzanas', 'pimientos', 'pescado']

print(compra)

compra.sort()
print(compra)

compra.sort(reverse = True)
print(compra)

# Ejemplo números:
totales = [234, 1, 543, 2345]

totales.sort()

print(totales)

totales.sort(reverse=True)

print(totales)
```

Al ejecutar todo el código te devuelve lo siguiente:

```
['yogures', 'manzanas', 'pimientos', 'pescado']
['manzanas', 'pescado', 'pimientos', 'yogures']
['yogures', 'pimientos', 'pescado', 'manzanas']
[1, 234, 543, 2345]
[2345, 543, 234, 1]
```

Función ordenada:

```
# Ejemplo con datos alfanuméricos:
compra = ['yogures', 'manzanas', 'pimientos', 'pescado']

print(compra)

lista_compra_ordenada = sorted(compra)
print(lista_compra_ordenada)

lista_compra_ordenada = sorted(compra, reverse=True)
print(lista_compra_ordenada)


# Ejemplo números:
totales = [234, 1, 543, 2345]

sorted_totales = sorted(totales)
print(sorted_totales)

sorted_totales = sorted(totales, reverse=True)
print(sorted_totales)
```

Resultados:

```
['yogures', 'manzanas', 'pimientos', 'pescado']
['manzanas', 'pescado', 'pimientos', 'yogures']
['yogures', 'pimientos', 'pescado', 'manzanas']
[1, 234, 543, 2345]
[2345, 543, 234, 1]
```

5.- ¿Qué es un operador de reasignación?

Se utiliza para poder añadir elementos a una tupla. Como ya sabemos, las tuplas son inmutables por lo que tendremos que crear otra variable con el mismo nombre para que se sobreescriba y esto es lo que se conoce como reasignación.

Ejemplo:
Esta es la tupla inicial:

```
menu = ('tabla de quesos', 'ensalada de bogavante', 'solomillo con
patatas')
print(menu)
```

Y si la ejecutamos nos devuelve:

```
('tabla de quesos', 'ensalada de bogavante', 'solomillo con patatas')
```

Queremos añadir un postre al menú: “torrijas con helado”. Para ello, vamos a crear una nueva variable llamada también “menu” y añadiremos dicho postre de la siguiente manera:

```
menu += ('torrijas con helado',)
print(menu)
```

Importante: poner la coma detrás del valor añadido. De lo contrario, nos dará error.

Tras imprimir, el resultado que nos devolverá será:

```
('tabla de quesos', 'ensalada de bogavante', 'solomillo con patatas',
'torrijas con helado')
```

La tabla de quesos son los entrantes, la ensalada el primer plato, el solomillo el segundo plato y las torrijas con helado el postre, es lo mismo que decir:

```
entrante, primer_plato, segundo_plato, postre = menu
```

Con lo cual podemos localizar cada elemento de manera individual de la siguiente manera:

```
print(entrante)
print(primer_plato)
print(segundo_plato)
print(postre)
```

Y nos devolverá:

```
tabla de quesos
ensalada de bogavante
solomillo con patatas
torrijas con helado
```

Código completo del ejemplo:

```
menu = ('tabla de quesos', 'ensalada de bogavante', 'solomillo con
patatas')

print(menu)

menu += ('torrijas con helado',)

print(menu)

entrante, primer_plato, segundo_plato, postre = menu

print(entrante)
print(primer_plato)
print(segundo_plato)
print(postre)
```

A través de la función id podemos comprobar que cada objeto, en este caso cada tupla, son distintas en la memoria del ordenador a pesar de haber utilizado el mismo nombre en las variables. Cada tupla tendrá un id distinto.
Ejemplo:

```
menu = ('tabla de quesos', 'ensalada de bogavante', 'solomillo con
patatas')

print(id(menu))

menu += ('torrijas con helado',)

print(id(menu))
```

El resultado que nos devuelve son 2 ids distintas:

```
2080951978624
2080952028352
```