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

La diferencia más importante entre listas y tuplas es su naturaleza mutable. Las listas son editables; puede cambiar, agregar o eliminar elementos después de crear la lista. Las tuplas son inmutables. Una vez creada una tupla, no se pueden cambiar sus elementos, agregar nuevos ni eliminar los existentes. Las listas se crean usando corchetes [] y las tuplas se crean usando paréntesis ().

```
list_numbers.append(6)  # ok
list_numbers[0] = 0  # ok

tuple_numbers.append(6)  # error
tuple_numbers[0] = 0  # error
```

2. ¿Cuál es el orden de las operaciones?

Las operaciones de mayor prioridad están en la parte superior, en la parte inferior, con baja prioridad. Los cálculos se realizan de izquierda a derecha, es decir, si una expresión contiene operadores de la misma precedencia, se ejecutará primero el de la izquierda.

El operador de exponenciación es una excepción a esta regla. De los dos operadores **, primero se ejecutará el derecho y luego el izquierdo.

() Soportes
** Exponenciación
+x, -x, ~x Unario más, menos y negación bit a bit
*, /, //, % Multiplicación, división, división entera, resto de división
+, - Suma y resta
≪, ≫ Cambios de bits
& poco y
^ OR exclusivo bit a bit (XOR)
| poco O
==, ≠, >, ≥, <, ≤, es, no está, en, no en Comparación, verificación de identidad, verificación de ocurrencia
no lógico NO
and lógico Y
or Lógico O

3. ¿Qué es un diccionario Python?

Los diccionarios en Python son colecciones desordenadas de objetos arbitrarios a los que se puede acceder mediante clave. Los diccionarios se utilizan principalmente cuando es necesario crear una estructura de datos flexible que permita realizar búsquedas rápidas. Los diccionarios también se utilizan como objetos de mapeo, que asignan valores a objetos arbitrarios. Los diccionarios son un par clave-valor. Las claves representan un objeto inmutable, mientras que el valor puede cambiar.

```
d = {'dict': 1, 'dictionary': 2}
d = dict(short='dict', long='dictionary')

list_one = ["one", "two", "three", "four", "five"]
list_two = [1, 2, 3, 4, 5]
```

```
the_dict = dict(zip(list_one, list_two))
```

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

Ordenar es el proceso de organizar elementos de una colección en un orden específico. En Python, puedes usar el método sort() para listas y la función sorted() para varios iterables para ordenar colecciones. Ambas funciones le permiten ordenar elementos en orden ascendente o descendente, según el parámetro especificado.
La función sorted() devuelve una nueva lista ordenada de la colección iterable que se le pasa. La función devuelve una lista ordenada sin cambiar el orden de los elementos de la colección original.

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
sorted_numbers = sorted(numbers)
```

El método sort() es un método integrado en Python que se utiliza para ordenar los elementos de una lista en su lugar, es decir, cambia la lista original sin crear una nueva lista.

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
numbers.sort()
```

El método sort() y la función sorted() también tienen un parámetro inverso adicional que se puede utilizar para ordenar colecciones en orden inverso.

5. ¿Qué es un operador de reasignación?

Las tuplas son inmutables, lo que significa que no se pueden cambiar, agregar ni eliminar elementos una vez creada la tupla.
Pero existen algunas soluciones.

```
my_tuple = (“apply”, “banan”, “cherry”)
my_list = list(my_tuple)
my_list[1] = "kiwi"
my)list.append("potato")
my_tuple = tuple(my_list)
my_tuple += (“cheese”)
```