1. **Advantage and Disadvantage of Generator** ?

**Advantages of Genetrators:**

Here is a summary of the advantages of generation expressions within python:

* Memory efficient method of generating sequence types in python.
* Adds further brevity and readability to written code. Generator expressions are [generator functions](https://wiki.python.org/moin/Generators) shortened.
* Time-efficient when compared to list comparisons.

**Disadvantage of Generators:**

if you need to access your data several times, you are still going to decrease your memory usage but your program will be slower since the generator and the data need to be generated each time. Therefore, it is generally not a good practice to use generator in this last scenario. The only disadvantage of generators is that they are not commonly used and are less popular.

**2.Normal vs Generator Function?**

**Difference Between Generator Function & Normal Function :**

* The main difference between a regular function and generator functions is that the state of generator functions are maintained through the use of the keyword yield and works much like using return, but it has some important differences.In generator functions, there are one or more yield functions, whereas, in Normal functions, there is only one function. When the generator function is called, the normal function pauses its execution, and the call is transferred to the generator function.

the difference is that yield saves the state of the function. The next time the function is called, execution continues from where it left off, with the same variable values it had before yielding, whereas the return statement terminates the function completely. Another difference is that generator functions don’t even run a function, it only creates and returns a generator object. Lastly, the code in generator functions only execute when **next()** is called on the generator object.

**3. Use cases of Generator?**

**Genator use of python:**

Python Generator functions allow you to declare a function that behaves likes an iterator, allowing programmers to make an iterator in a fast, easy, and clean way. An iterator is an object that can be iterated or looped upon. It is used to abstract a container of data to make it behave like an iterable object. Examples of iterable objects that are used more commonly include lists, dictionaries, and strings.