Generators in python

Generator in Python are special type of functions that allow you to create an sequence of values. A generator function returns a generator object, which can be used to generate the values one by one as you iterate over it. Generators are a powerful tool for working with the values on the fly, rather than having to create and store the entire sequence in memory.

Creating a Generator

In Python, you can create a generator by using the yield statement in a function. The yield statement returns a value from the generator and suspends the execution of the function until the next value is requested. Here's an example.

```
def my_Generator():
    for i in range(10):
        yield i

gen=my_Generator()
# print(next(gen))
# print(next(gen))
# print(next(gen))

for j in gen:
    print(j)
```

Benefits of Generators

Generators offer several benefits over other types if sequence such as list, tuples and sets. One of the main benefits of generators is that they allow you to generate the values on the fly, rather than having to create and store the entire sequence in memory. This makes generators a powerful tool for working with large or complex data sets, as you can generate the values as you need them, rather having to store them all in memory at once.

Another benefit of generator is that they are lazy which means that the values are generated only when they are requested. This allows you to generate the values in a more efficient and memory friendly manner, as you don’t have to generate all the values up front.

Source Code

```
def my_Generator():
    for i in range(10):
        yield i

gen=my_Generator()
# print(next(gen))
# print(next(gen))
# print(next(gen))

for j in gen:
    print(j)
```

Thank You