Full Stack Development
Sesi 6

ADVANCED PYTHON

Iterators

An iterator is an object that can be iterated upon which means that you can traverse through all the values. List, tuples, dictionaries, and sets are all iterable objects.

To create an object as an iterator you have to implement the methods __iter__() and __next__() to your object where __iter__() returns the iterator object itself. This is used in for and in statements.

__next__() method returns the next value in the sequence. In order to avoid the iteration to go on forever, raise the StopIteration exception.

Why use iterators?

Iterators allow us to create and work with lazy iterable which means you can use an iterator for the lazy evaluation. This allows you to get the next element in the list without re-calculating all of the previous elements. Iterators can save us a lot of memory and CPU time.

Generators

Generator functions act just like regular functions with just one difference that they use the Python yieldkeyword instead of return . A generator function is a function that returns an iterator. A generator expression is an expression that returns an iterator. Generator objects are used either by calling the next method on the generator object or using the generator object in a “for in” loop.

Decorator

A decorator in Python is any callable Python object that is used to modify a function or a class. It takes in a function, adds some functionality, and returns it. Decorators are a very powerful and useful tool in Python since it allows programmers to modify/control the behavior of function or class. Decorators are usually called before the definition of a function you want to decorate.

There are two different kinds of decorators in Python:

- Function decorators
- Class decorators

Python Introduction Sesi 3

External References

Adv. Python - Visit Here