Generators

Decorators

- We've learned how to create functions with def and the return statement.
- Generator functions allow us to write a function that can send back a value and then later resume to pick up where it left off.

Decorators

- This type of function is a generator in Python, allowing us to generate a sequence of values over time.
- The main difference in syntax will be the use of a yield statement.

Decorators

- When a generator function is compiled they become an object that supports an iteration protocol.
- That means when they are called in your code they don't actually return a value and then exit.

Decorators

- Generator functions will automatically suspend and resume their execution and state around the last point of value generation.
- The advantage is that instead of having to compute an entire series of values up front, the generator computes one value waits until the next value is called for.

Decorators

- For example, the range() function doesn’t produce an list in memory for all the values from start to stop.
- Instead it just keeps track of the last number and the step size, to provide a flow of numbers.

Decorators

- If a user did need the list, they have to transform the generator to a list with **list(range(0,10))**
- Let’s explore how to create our own generators!

Iterators and Generators Homework Solutions

PIERIAN DATA