BME646 and ECE60146: Homework 1

# Mykhailo Tsysin

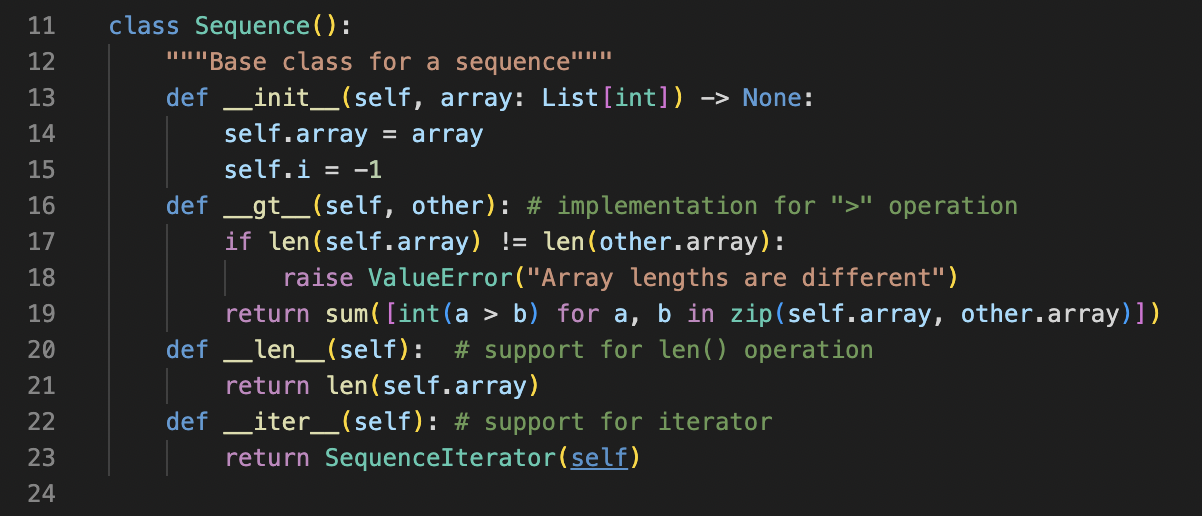
# Jan 15 2023

In this homework, we will experiment with object-oriented design in Python. The development process with code snippets is described here. For full .py solution, please look at corresponding file.

# Programming tasks:

## Create sequence class:

The class definition is straightforward:



For now, we’re only interested in basic definition and \_\_init\_\_ method which declares an array. Other parts will be described later.

## Create Fibonacci class and make it callable:

The Fibonacci class will look like this:

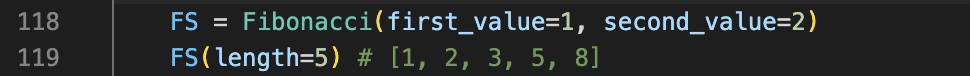
Text

Description automatically generated

The small improvement we can make right away is to notice that if we have some portion of numbers already stored in array, we don’t need to compute them again, otherwise the solution is straightforward. This will be out \_\_call\_\_ method.

The results for the driver code with the reproduced results from the problem statement are presented below:

Input:



Ouput:



Other tests:

Text

Description automatically generated

Output:

A picture containing text

Description automatically generated

## Add iterator to the class

This can be accomplished as shown on the class slides. We create another iterator class for Sequence class that can be referenced in Sequence. The complete picture looks like this:

Text

Description automatically generated

When we call Sequence as iterator, SequenceIterator instance is returned with the logic like what’s shown on the slides. Separate definition on next method is not needed for Python3. We alse add \_\_len\_\_ method for len() function.

The provided and custom test case is shown:

A screenshot of a computer

Description automatically generated with medium confidence

Output:

A picture containing text

Description automatically generated

## Create Prime class

Prime class has similar logic to Fibonacci class. We also reuse array form previous run. Prime numbers are generated by checking all the numbers after the last existing prime in the array and adding them if they’re not divisible by any pf the existing primes. The process continues until we have enough numbers. The code looks like this:

Text

Description automatically generated

The test case from the problem statement:

Text

Description automatically generated

Output:

Text

Description automatically generated

Also:

Input

A screenshot of a computer

Description automatically generated with medium confidence

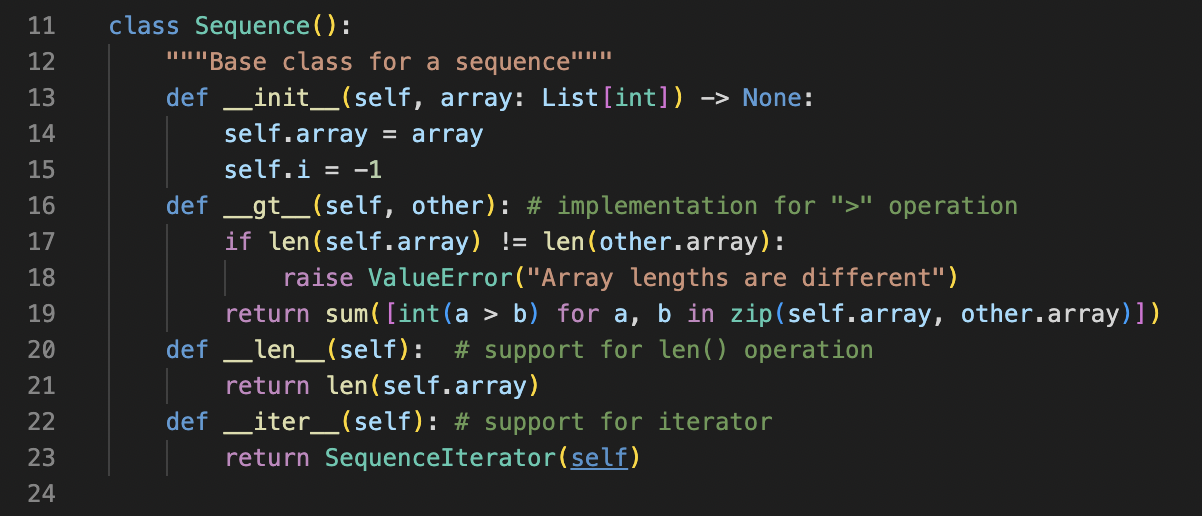
Output

Graphical user interface

Description automatically generated with low confidence

## Add “>” operation:

This is accomplished by adding \_\_gt\_\_() method to the base class, as shown on the previous picture and duplicated here:



If lengths are different, we raise an error. If not, we count positions where element from first array is greater than element form second array.

Test case:

Text

Description automatically generated

Output:

Text

Description automatically generated