Homework 0x02 - Practicing with Generators

This short homework assignment is to be completed individually and will give you some practice in writing simple Python code. It will be due on Monday, November 6th before 11:10 AM.

The Assignment

Write a Python program for a PC (not a MicroPython board) that accomplishes the following objectives listed below. Your program will be run and evaluated by the grader and must be free of errors.

1. Write and test a generator function that can produce Fibonacci numbers sequentially. Recall that the Fibonacci sequence is defined by the following recursive relationship.

   $$f_0 = 0; \qquad f_1 = 1; \qquad f_n = f_{n-1} + f_{n-2}$$

   Your generator function must take an input parameter representing the length of the sequence. The input parameter should be validated by your code to make sure that it is of the appropriate type and value. Use the Python `raise` statement with `TypeError` and `ValueError` for invalid inputs. See here for details: https://docs.python.org/3/tutorial/errors.html#raising-exceptions.

2. Test your generator function by making generator objects from the function for various sequence lengths that are both valid and invalid. In order for the grader to be able to run your code without error you will need to use a `try/except` block so that the error can be caught without crashing your program. See here for details: https://docs.python.org/3/tutorial/errors.html#handling-exceptions.

   Show that the generator produces the correct sequence for each length by printing out the full sequence.

3. Write a block of code that will add up every third Fibonacci number. That is, write a block of code that will compute the sum $f_0 + f_3 + f_6 + \cdots + f_n$ for all values of n that are multiples of 3.

   This block of code must meet the following requirements:

   - The summation code must use the Fibonacci generator function exactly as written for the preceding objectives. That is, do not modify the generator function to perform this summation.
   - Your code may not, at any time during execution, store the sequence of Fibonacci numbers in memory. That is, you must work with each Fibonacci number as produced by your generator without storing those Fibonacci numbers for later use.
   - Your program must be able to fully execute in less than 1 second for sequences up to a length of 100 thousand Fibonacci numbers.

Restrictions

Please avoid using google searches for this assignment. While, in general, we fully support proper and ethical code reuse in ME 405, the objectives for this assignment are short enough that too much googling will almost entirely eliminate the learning objectives.

For example, the Fibonacci generator function that you must write only needs to be about 6 or 8 lines of code (excluding comments). Please use the generator coding example from lecture and make your own modifications to achieve the assignment objectives.

Deliverables

You may appreciate using a tool such as Jupyter notebook or Jupyter lab to complete this assignment, but other environments like Spyder, PyCharm, or VS Code will work fine too.

Your code should be set up as follows:

- Write your generator function near the top of your Python file; you shouldn't need any special imports for this assignment aside from the `time` module, but they may go above your function.
- Write your testing code and the summation code within an `if __name__ == "__main__":` block at the bottom of the Python file.
  - Test your generator function by itself by printing a reasonably sized sequence to the console window.
  - Show that your generator function raises the appropriate exceptions by attempting to instantiate generator objects using invalid sequence lengths. Use the exception handling techniques mentioned in lecture to catch any exceptions that occur without them causing your program to fail or exit early.

    While it is considered out of the scope of the assignment, you may experiment with purpose-built tools for testing Python code. The `unittest` module may worth investigating if you want a more "Pythonic" way to test your code.
  - Use `time.perf_counter()` or `time.perf_counter_ns()` to measure the duration of time it takes to perform your summation of every third out of the first 100 thousand Fibonacci numbers. *Do not* print any values while performing this benchmark or you will end up measuring the duration of the print function instead of your own code.
  - Print the result of the summation (both the final summed value and the duration of time) to the console window so that the grader can make sure your math checks out properly and you've met the assignment requirements.

When you have completed the assignment, upload the Python source code to Canvas.

Extra Credit Opportunity

For a 30% bonus to your assignment grade, complete the summation aspect of the assignment, objective #3, above, *without using a loop to complete the summation* with Python tools like `map()`, `filter()`, and `sum()`. You may still use a loop within the generator function though.