F. Wermelinger
Office: Pierce 211

Pair-Programming 10

Python generators

Issued:	November 4, 2022
Due:	November 18, 2022 11:59pm

In this pair-programming session you will implement a Python generator that can be used as an iterator for an object.

You should work on the exercises in groups of 3 to 4 students via a `tmate` session. Your team members can submit the same file. Please indicate your names in a header in the files. See the tutorials on the class website for an example pair-programming workflow.[1] Do not forget to commit and push your work when you are done. Ensure that you are on your *default branch* for this and not, possibly, on your homework branch.

Exercise 1: Python Generators

Deliverables:

1. `exercise_1.py`

The following code models a text type that allows to iterate over words, where the `re` module of the standard library is used to find words in an input string.[2]

```
import re

class Text:
    """Text representation with word iteration support.

    Example:
    --------
    >>> text = Text("Hello CS107/AC207 class!")
    >>> list(text)
    ['Hello', 'CS107', 'AC207', 'class']

    """
```

[1] https://harvard-iacs.github.io/2022-CS107/pages/tutorials.html#tutorial-pp
[2] https://docs.python.org/3/library/re.html

```
    def __init__(self, text):
        self.text = text

    def __iter__(self):
        return WordIterator(self.text)

class WordIterator:
    """Iterate of words in input text."""

    def __init__(self, text):
        self.words = re.findall(r'\b[a-zA-Z0-9]+\b', text)

    def __next__(self):
        try:
            return self.words.pop(0)
        except IndexError:
            raise StopIteration

    def __iter__(self):
        return self
```

Please implement the `WordIterator` in the `Text` class directly using a Python generator function. You can find the skeleton code for this exercise in `lab/pp10/data/exercise_1.py`.