F. Wermelinger
Office: Pierce 211

Pair-Programming 9

Python iterators, binary search trees

Issued:	October 28, 2022
Due:	November 11, 2022 11:59pm

In this pair-programming session you will create a simple iterator for a Fibonacci sequence and explore the construction of a binary search tree.

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 Iterators

Deliverables:

1. `exercise_1.py`

Fibonacci numbers are defined by the recurrence relation

$$F_n = F_{n-1} + F_{n-2} \quad \text{for} \quad n > 1 \tag{1}$$

with $F_0 = 0$ and $F_1 = 1$.

We start with the following class to describe `N` Fibonacci numbers:

```python
class Fibonacci:

    def __init__(self, N):
        self.N = N

    def __iter__(self):
        return FibonacciIterator(self.N)
```

[1] https://harvard-iacs.github.io/2022-CS107/pages/tutorials.html#tutorial-pp

The `__iter__` special method returns an *iterator* which is an object expected to be returned by an iterable[2] when it is passed as an argument to the `iter()` built-in function.[3] All sequence objects like lists or tuples are iterable. Other objects can be implemented as iterables by implementing the `__iter__` special method.

The `FibonacciIterator` class should iterate over F_n starting with $n = 0$ and stop when iterated `N` times. An application of `Fibonacci` could look like this:

```
>>> fib = Fibonacci(12)
>>> iter(fib)
<exercise_1.FibonacciIterator object at 0x7fd9b2ec79d0>
>>> len(list(fib))
12
>>> list(fib) # the list() built-in assumes the object `fib` is iterable
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Please implement the `FibonacciIterator` class based on the recurrence relation given in equation 1. You can find the skeleton code in `lab/pp9/data/exercise_1.py`. See the docstring of the `FibonacciIterator` class for the requirements.

Please see `solution/exercise_1.py` for the solution code.

Exercise 2: Binary Search Tree

Deliverables:

1. `exercise_2.png` (or `exercise_2.pdf`)

In homework 5 and 6 we will work with binary search trees (BST). The first part will be concerned with the construction of such trees and the following part will extend the algorithm for deleting nodes in a tree.

In the following you will simply draw (construct) a BST based on some given values. Assume you are given the unsorted array

$$[13, 7, 19, 17, 3, 29, 5, 31, 2, 11].$$

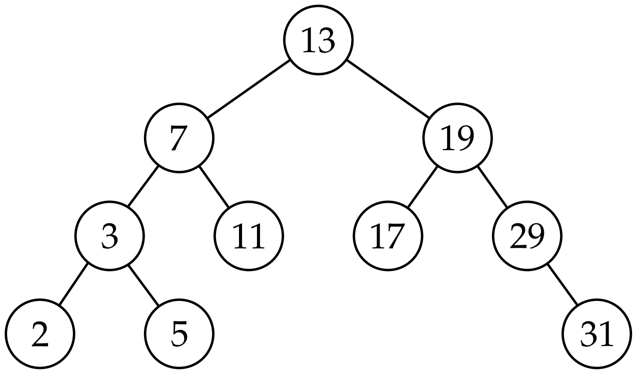
You construct a binary search tree by picking the first number in the array and place it at the *root* of you tree. This will be the root node. Take the next number in the array and start at the root. If the number is *less* than the value in the root node go to the *left* child node, otherwise to the right child node and repeat the comparison. If the child does not exist, the number you are currently comparing becomes the value of a new child node in this place.

Given the array of numbers above, please draw the corresponding BST. You may take a picture if you draw it by hand and save it in the file `exercise_2.png` or save it in `exercise_2.pdf` if you draw it digitally.

[2] https://docs.python.org/3/glossary.html#term-iterable
[3] https://docs.python.org/3/library/functions.html#iter

If you delete the root node 13, you would end up with two disjoint trees. What could you do to replace the deleted root node with a new node? Please write down options you can think of to replace the deleted root node with another node in the tree (2 to 3 sentences).

The binary search tree is shown below. If you remove the root node you end up with two disjoint trees. You can replace the deleted root node with either the largest node in the left subtree (11) *or* the smallest node in the right subtree (17).


13
7
19
3
11
17
29
2
5
31