hw1_ChengjunGuo

Chengjun Guo

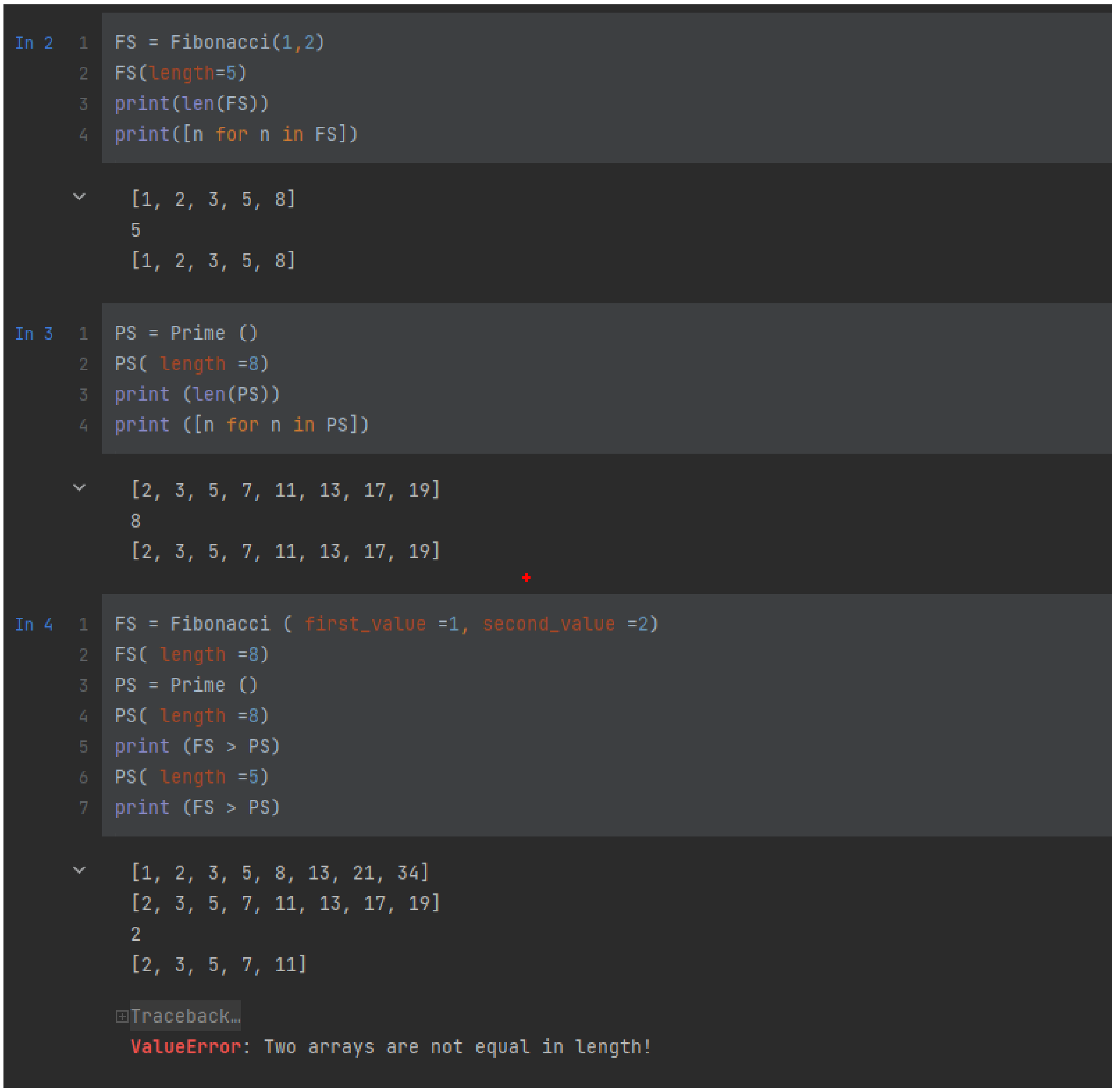
January 2023

1 Introduction

Based on the requirement, I created three classes. The base called Sequence and two subclasses called Prime and Fibonacci. Fibonacci's callable is easy to keep track because it's adding the stored previous two values to extend. There's no way to do the same to the Prime class. I'm storing a list of prime numbers in the database. Whenever the callable is called, it will take the list with expected length.

2 Output

2.1 Reproduction



```
In 2
FS = Fibonacci(1,2)
FS(length=5)
print(len(FS))
print([n for n in FS])
```

```
[1, 2, 3, 5, 8]
5
[1, 2, 3, 5, 8]
```

```
In 3
PS = Prime ()
PS( length =8)
print (len(PS))
print ([n for n in PS])
```

```
[2, 3, 5, 7, 11, 13, 17, 19]
8
[2, 3, 5, 7, 11, 13, 17, 19]
```

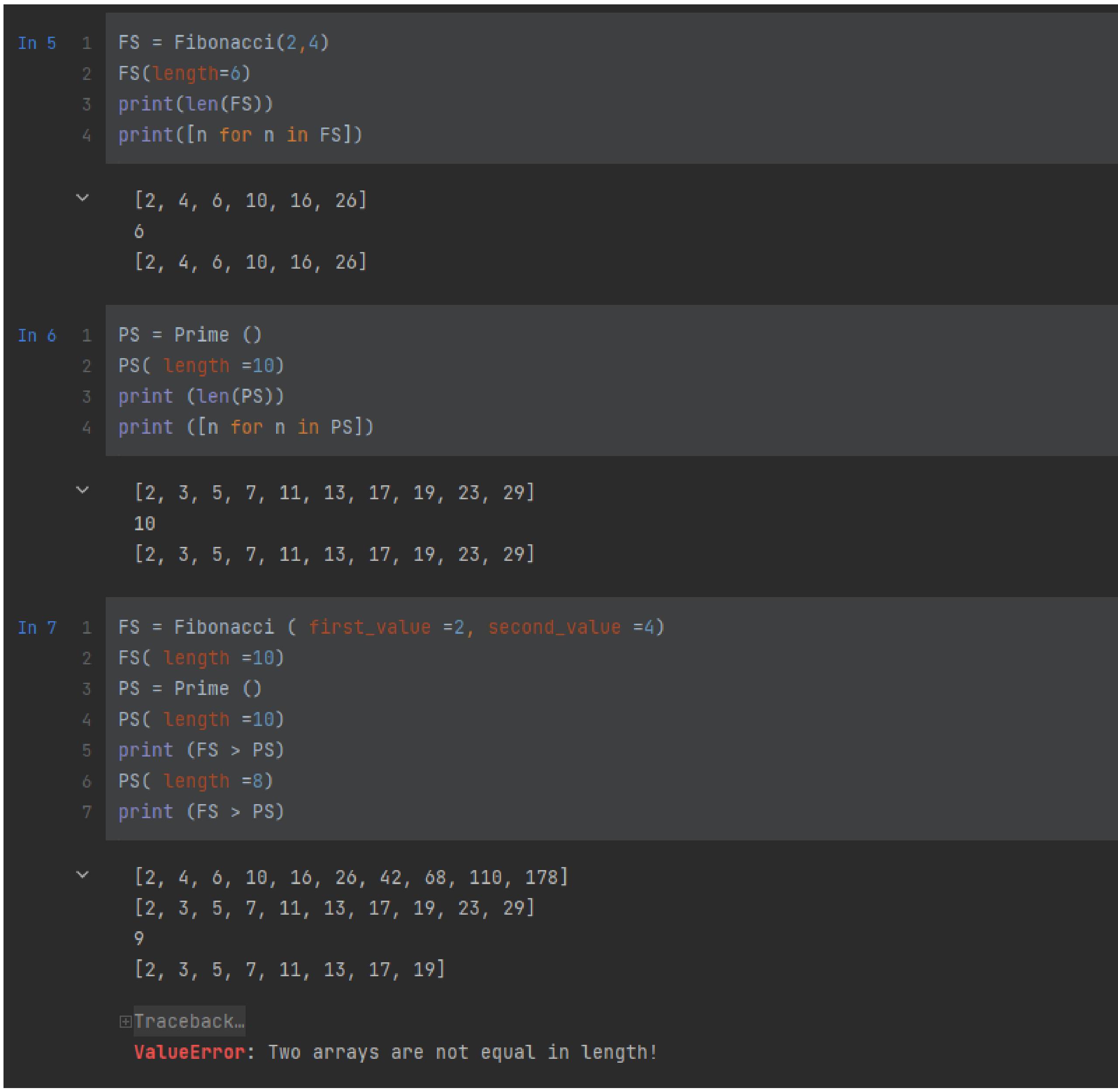
```
In 4
FS = Fibonacci ( first_value =1, second_value =2)
FS( length =8)
PS = Prime ()
PS( length =8)
print (FS > PS)
PS( length =5)
print (FS > PS)
```

```
[1, 2, 3, 5, 8, 13, 21, 34]
[2, 3, 5, 7, 11, 13, 17, 19]
2
[2, 3, 5, 7, 11]

⊞Traceback…
ValueError: Two arrays are not equal in length!
```

Figure 1: reproduction

2.2 My input



```
In 5
FS = Fibonacci(2,4)
FS(length=6)
print(len(FS))
print([n for n in FS])
```

```
[2, 4, 6, 10, 16, 26]
6
[2, 4, 6, 10, 16, 26]
```

```
In 6
PS = Prime ()
PS( length =10)
print (len(PS))
print ([n for n in PS])
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
10
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

```
In 7
FS = Fibonacci ( first_value =2, second_value =4)
FS( length =10)
PS = Prime ()
PS( length =10)
print (FS > PS)
PS( length =8)
print (FS > PS)
```

```
[2, 4, 6, 10, 16, 26, 42, 68, 110, 178]
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
9
[2, 3, 5, 7, 11, 13, 17, 19]

⊞Traceback…
ValueError: Two arrays are not equal in length!
```

Figure 2: my input

3 Source code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np

class Sequence(object):
    def __init__(self,array):
        self.array = array
    def __next__(self):
        if self.index < self.__len__():
            result = self.array[self.index]
            self.index += 1
            return result
        else:
            raise StopIteration
    def __iter__(self):
        self.index = 0
        return self
    def __len__(self):
        return len(self.array)
    def __gt__(self, other):
        if self.__len__() != other.__len__():
            raise ValueError('Two arrays are not equal in
                length!')
        else:
            l1 = np.array(self.array)
            l2 = np.array(other.array)
            result = l1 > l2
            return result.sum()

class Fibonacci(Sequence):
    def __init__(self,first_value,second_value):
        array = [first_value,second_value]
        super().__init__(array)
    def __call__(self,length):
        first = self.array[0]
        second = self.array[1]
        new_array = []
        for i in range(length):
            if i==0:
                new_array.append(first)
            elif i==1:
                new_array.append(second)
            else:
                new_array.append(new_array[i-1]+new_array
                    [i-2])
        self.array = new_array
        print(self.array)

class Prime(Sequence):
    def __init__(self):
        super().__init__([])
        # reference: https://www.reddit.com/r/Python/
            comments/cvmhaj/
            find_first_n_prime_numbers_using_numpy_100x_faster
            /
        n=1000000
        s = np.arange(3, n, 2)
        for m in range(3, int(n ** 0.5)+1, 2):
            if s[(m-3)//2]:
                s[(m*m-3)//2::m]=0
        self.database = np.r_[2, s[s>0]]
    def __call__(self, length):
        self.array = self.database[:length].tolist()
        print(self.array)


# In[5]:


FS = Fibonacci(2,4)
FS(length=6)
print(len(FS))
print([n for n in FS])


# In[6]:


PS = Prime ()
PS( length =10)
print (len(PS))
print ([n for n in PS])
```

```
# In[7]:


FS = Fibonacci ( first_value =2, second_value =4)
FS( length =10)
PS = Prime ()
PS( length =10)
print (FS > PS)
PS( length =8)
print (FS > PS)


# In[ ]:
```