ECE 60146 HW1 Report

Zhengxin Jiang
(jiang839@purdue.edu)

January 17, 2023

1 Run Results of Given Parameters

Task 1-4

In [54]:
```
# Task 1-4
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]
```

Figure 1: Reproduction of task 1-4

Task 5

In [55]:
```
# Task 5
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]
```

Figure 2: Reproduction of task 5

Task 6

```
In [56]:
# Task 6
FS = Fibonacci(1,2)
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]

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17632/2518245587.py in <module>
      5 print(FS>PS)
      6 PS(length = 5)
----> 7 print(FS>PS)

~\AppData\Local\Temp/ipykernel_17632/955248393.py in __gt__(self, other)
     25         # check if two objects have the same length
     26         if len(self.array) != len(other.array):
---> 27             raise ValueError('Two arrays are not equal in length !')
     28
     29         count = 0

ValueError: Two arrays are not equal in length !
```

Figure 3: Reproduction of task 6

2 Run Results of Self-chosen Parameters

For the following runs, the first two numbers of the Fibonacci class object and the length of both class objects are changed.

Task 1-4

```
In [58]:
# Task 1-4
FS = Fibonacci(3,5)
FS(length = 6)
print(len(FS))
print([n for n in FS])
```

```
[3, 5, 8, 13, 21, 34]
6
[3, 5, 8, 13, 21, 34]
```

Figure 4: Result of task 1-4 with changed parameters

Task 5

```
In [59]:
1 # Task 5
2 PS = Prime()
3 PS(length = 10)
4 print(len(PS))
5 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]
```

Figure 5: Result of task 5 with changed parameters

Task 6

```
In [60]:
1 # Task 6
2 FS = Fibonacci(3,5)
3 FS(length = 8)
4 PS = Prime()
5 PS(length = 8)
6 print(FS>PS)
7 PS(length = 10)
8 print(FS>PS)
```

```
[3, 5, 8, 13, 21, 34, 55, 89]
[2, 3, 5, 7, 11, 13, 17, 19]
8
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_17632/2518583624.py in <module>
      6 print(FS>PS)
      7 PS(length = 10)
----> 8 print(FS>PS)

~\AppData\Local\Temp/ipykernel_17632/955248393.py in __gt__(self, other)
     25         # check if two objects have the same length
     26         if len(self.array) != len(other.array):
---> 27             raise ValueError('Two arrays are not equal in length !')
     28
     29         count = 0

ValueError: Two arrays are not equal in length !
```

Figure 6: Result of task 6 with changed parameters

3 Source code

```
# ECE60146 HW1
# Zhengxin Jiang
# jiang839

class Sequence(object):
    def __init__(self, array):
        self.array = array

    def __iter__(self):
        self.idx = -1
        return self

    def __next__(self):
        self.idx += 1

        if self.idx<len(self.array):
            return self.array[self.idx]
        else:
            raise StopIteration

    def __len__(self):
         return len(self.array)

    def __gt__(self, other):
        # check if two objects have the same length
        if len(self.array) != len(other.array):
            raise ValueError('Two␣arrays␣are␣not␣equal␣in␣length␣!')

        count = 0
        for i in range(len(self.array)):
            if self.array[i] > other.array[i]:
                count += 1

        return count


# subclass
class Fibonacci(Sequence):

    def __init__(self, first_value, second_value):
        Sequence.__init__(self, [first_value, second_value])


    def __call__(self, length):
        # init
        self.array = self.array[:2]

        if length > 2:
            for i in range(2, length):
                self.array.append(self.array[i-1]+self.array[i-2])

        print(self.array)
```

```
# subclass
class Prime(Sequence):

    def __init__(self):
        Sequence.__init__(self, [])

    def __call__(self, length):
        # init
        self.array = []

        if length == 1:
            self.array.append(2)

        if length > 1:
            self.array.append(2)

            num = 3
            while len(self.array)<length:
                is_prime = True

                # test if a number is prime
                for i in range(2, num):
                    if num%i == 0:
                        is_prime = False
                        break

                if is_prime:
                    self.array.append(num)

                num += 1

        print(self.array)


# Main
# Task 1-4
FS = Fibonacci(1,2)
FS(length = 5)
print(len(FS))
print([n for n in FS])

# Task 5
PS = Prime()
PS(length = 8)
print(len(PS))
print([n for n in PS])

# Task 6
FS = Fibonacci(1,2)
FS(length = 8)
PS = Prime()
PS(length = 8)
print(FS>PS)
```

```
PS(length = 5)
print(FS>PS)
```