Homework 1

Alexandre Olivé Pellicer

1. I create the class as mentioned in the assignment

In [ ]:
```
class Sequence (object):
    def __init__ (self, array):
        self.array = array
```

2. I create Arithmetic as a subclass of Sequence with the two input parameters specified in the assignment

In [ ]:
```
class Arithmetic (Sequence):
    def __init__(self, start, step):
        self.start = start
        self.step = step
```

3. In order to make the instances of "Arithmetic" callable, I define the method __call__. When calling this method, an array of length "length" starting from "self.start" with step "self.step" will be stored in self.array. To do it, I use the methods "list" to create the array and "range" which returns a sequence of numbers by specifying the first number of the sequence, the last number of the sequence and the step. Finally, I use the print function to print the array.

In [ ]:
```
class Arithmetic (Sequence):
    def __init__(self, start, step):
        self.start = start
        self.step = step

    def __call__(self, length):
        self.array = list(range(self.start, self.start+length*self.step, self.step)
        print(self.array)
```

In order to see that the code is well implemented I reproduce the snippet from the assignment and others

In [ ]:
```
AS = Arithmetic (start =1 , step =2)
AS(length =5)
```

```
[1, 3, 5, 7, 9]
```

In [ ]:
```
AS = Arithmetic (start =3 , step =3)
AS(length =4)
```

```
[3, 6, 9, 12]
```

In [ ]:
```
AS = Arithmetic (start =0 , step =10)
AS(length =0)
```

```
[]
```

In [ ]:
```
AS = Arithmetic (start =3 , step =1)
AS(length =10)
```

```
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

4. I add the __iter__ method in the superclass Sequence. As shown in the slides from class, the iterator will be a class that provides implementation for the method __next__. In my case, it is the Iterator class. Every time the __next__ method is called, it returns the next element of the array until reaching the end. This is done by increasing the value of self.index every time and accessing the position self.index in the array.

In [ ]:
```
class Sequence (object):
    def __init__ (self, array):
        self.array = array

    def __iter__(self):
        return Iterator(self)

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

class Iterator:
    def __init__(self, seq):
        self.items = seq.array
        self.index = -1
    def __iter__(self):
        return self
    def __next__(self):
        self.index += 1
        if self.index < len(self.items):
            return self.items[self.index]
        else:
            raise StopIteration

class Arithmetic (Sequence):
    def __init__(self, start, step):
        self.start = start
        self.step = step

    def __call__(self, length):
        self.array = list(range(self.start, self.start+length*self.step, self.step)
        #print(self.array)
```

In order to see that the code is well implemented I reproduce the snippet from the assignment and others

In [ ]:
```
AS = Arithmetic( start =1 , step =2 )
AS(length =5)
print(len(AS))
print([n for n in AS])
```

```
5
[1, 3, 5, 7, 9]
```

In [ ]:
```
AS = Arithmetic (start =3 , step =3)
AS(length =4)
print(len(AS))
print([n for n in AS])
```

```
4
[3, 6, 9, 12]
```

In [ ]:
```
AS = Arithmetic (start =0 , step =10)
AS(length =0)
print(len(AS))
print([n for n in AS])
```

```
0
[]
```

In [ ]:
```
AS = Arithmetic (start =3 , step =1)
AS(length =10)
print(len(AS))
print([n for n in AS])
```

```
10
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

5. I copy the structure of the Arithmetic class to create the Geometric class. I just change the way of creating the array that is stored in self.array once the start value, ratio and length are specified in order to create a geometric sequence instead of an arithmetic sequence.

In [ ]:
```
class Geometric (Sequence):
    def __init__(self, start, ratio):
        self.start = start
        self.ratio = ratio

    def __call__(self, length):
        self.array = [self.start * (self.ratio ** i) for i in range(length)]
        #print(self.array)
```

In order to see that the code is well implemented I reproduce the snippet from the assignment and others

In [ ]:
```
GS = Geometric( start =1 , ratio =2 )
GS(length =8)
print(len( GS ))
print([n for n in GS])
```

```
8
[1, 2, 4, 8, 16, 32, 64, 128]
```

In [ ]:
```
GS = Geometric( start =3 , ratio =4 )
GS(length =5)
print(len( GS ))
print([n for n in GS])
```

```
5
[3, 12, 48, 192, 768]
```

In [ ]:
```
GS = Geometric( start =5 , ratio =3 )
GS(length =2)
print(len( GS ))
print([n for n in GS])
```

```
2
[5, 15]
```

6. In this case I define the method __eq__ in the superclass Sequence. This method will be called when using the operator ==. The arguments of the method __eq__ are "self" and "other". "self" refers to the object in the left side of the operator == and "other" refers to the object in the right side of the operator ==. Using the method __len__ previously defined, I first compare that the lengths are the same. If not, I throw a ValueError exception. Otherwise, equal_count stores the number of elements that are equal element_wise. I use "for a, b in zip(self, other)" to go through the elements of both arrays element-wise calling the __iter__ method previously defined. "if a == b" I generate "1" for the counter. Finally, all the 1s are summed.

In [ ]:
```
class Sequence (object):
    def __init__ (self, array):
        self.array = array

    def __iter__(self):
        return Iterator(self)

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

    def __eq__(self, other):
        if len(self) != len(other):
            raise ValueError("Two arrays are not equal in length !")
        equal_count = sum(1 for a, b in zip(self, other) if a == b)
        return equal_count

class Iterator:
    def __init__(self, seq):
        self.items = seq.array
        self.index = -1
    def __iter__(self):
        return self
    def __next__(self):
        self.index += 1
        if self.index < len(self.items):
            return self.items[self.index]
        else:
            raise StopIteration

class Arithmetic (Sequence):
    def __init__(self, start, step):
        self.start = start
        self.step = step

    def __call__(self, length):
        self.array = list(range(self.start, self.start+length*self.step, self.step)
        #print(self.array)

class Geometric (Sequence):
    def __init__(self, start, ratio):
        self.start = start
        self.ratio = ratio

    def __call__(self, length):
        self.array = [self.start * (self.ratio ** i) for i in range(length)]
        #print(self.array)
```

In order to see that the code is well implemented I reproduce the snippet from the assignment and others.

```
In [ ]: AS = Arithmetic ( start =1 , step =2 )
        AS( length =5 ) # [1, 3, 5, 7, 9]
        GS = Geometric ( start =1 , ratio =2 )
        GS( length =5 ) # [1, 2, 4, 8, 16]
        print(AS == GS ) # 1
        GS( length =8 ) # [1, 2, 4, 8, 16 , 32 , 64 , 128]
        print( AS == GS ) # will raise an error
```

```
1
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/home/aolivepe/ECE60146/HW1/task.ipynb Cell 29 line 7
      <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoli
vepe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=4'>5</a> print(AS ==
GS ) # 1
      <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoli
vepe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=5'>6</a> GS( length =
8 ) # [1, 2, 4, 8, 16 , 32 , 64 , 128]
----> <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoli
vepe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=6'>7</a> print( AS ==
GS ) # will raise an error

/home/aolivepe/ECE60146/HW1/task.ipynb Cell 29 line 1
     <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoliv
epe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=10'>11</a> def __eq__
(self, other):
     <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoliv
epe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=11'>12</a>     if len
(self) != len(other):
---> <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoliv
epe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=12'>13</a>         rai
se ValueError("Two arrays are not equal in length !")
     <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoliv
epe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=13'>14</a>     equal_c
ount = sum(1 for a, b in zip(self, other) if a == b)
     <a href='vscode-notebook-cell://ssh-remote%2Bsemafor3.ecn.purdue.edu/home/aoliv
epe/ECE60146/HW1/task.ipynb#X40sdnNjb2RlLXJlbW90ZQ%3D%3D?line=14'>15</a>     return
equal_count

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

```
In [ ]: AS = Arithmetic ( start =1 , step =1 )
        AS( length =5 )
        print([n for n in AS])

        GS = Geometric ( start =1 , ratio =2 )
        GS( length =5 )
        print([n for n in GS])

        print(AS == GS )
```

```
[1, 2, 3, 4, 5]
[1, 2, 4, 8, 16]
2
```