

ECE 601: Homework 1
Wei Xu
Email: xu1639@purdue.edu
Due date: 11:59 PM, Jan. 15, 2024
(Spring 2024)

1 Implementation Description

1.1 Task 1

The class `Sequence` is created and initialized using the `__init__` method with the input parameter `array` as the object property.

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

1.2 Task 2

Inheriting the class `Sequence`, the subclass `Arithmetic` is created. It is initialized using the `__init__` method with two input parameters `start` and `step`. To inherit from the parent class, the `super()` function is used.

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

1.3 Task 3

the `__call__` method is added to the class `Arithmetic` to make its instances behave like functions and can be called like a function. A `for` loop is used to generate the sequence. Its length is controlled by the input parameter `length`. The generated sequence is stored by the variable `array`. The `print` function ensures the generated sequence is printed.

```
class Arithmetic(Sequence):  
    ...  
    def __call__(self, length):  
        temp = self.start  
        self.array = []  
        for _ in range(length):  
            self.array.append(temp)  
            temp += self.step  
        print(self.array) # Print after generating
```

1.4 Task 4

To make the instance of the class `Sequence` be used as an iterator, the `__iter__` and `__next__` methods need to be implemented. The `__iter__` method initializes the `index` variable and returns

the iterator object. The `__next__` method returns the next element in the sequence. The length of the sequence can be obtained through the `__len__` method. By adding the `StopIteration` statement, it can be limited to a certain number (the sequence length) of iterations.

```
class Sequence(object):
    ...
    def __len__(self):
        return len(self.array)
    def __iter__(self):
        self.index = -1
        return self
    def __next__(self):
        self.index += 1
        if self.index < len(self):
            return self.array[self.index]
        else:
            raise StopIteration
```

1.5 Task 5

Inheriting the class `Sequence`, the subclass `Geometric` is created. Similar to the subclass `Arithmetic`, it is initialized using the `__init__` method with two input parameters `start` and `ratio`. The `__call__` method needs to be modified to accommodate the geometric sequence. Because it inherits the class `Sequence`, it can be used as an iterator without further modifications.

```
class Geometric(Sequence):
    def __init__(self, start, ratio):
        super().__init__([])
        self.start = start
        self.step = ratio
    def __call__(self, length):
        temp = self.start
        self.array = []
        for _ in range(length):
            self.array.append(temp)
            temp *= self.step
        print(self.array) # Print after generating
```

1.6 Task 6

The `__eq__` method is added to support the "equal" comparison. A `for` loop is applied to element-wise comparison. The number of identical elements is maintained by the variable `eq_count`, which is returned finally. In addition, the `ValueError` is thrown if the lengths of the two sequences are not equal.

```
class Sequence(object):
    ...
    def __eq__(self, other):
        if len(self) != len(other):
            raise ValueError('Two arrays are not equal in length!')
        else:
            eq_count = 0
            for i in range(len(self)):
                if self.array[i] == other.array[i]:
```

```
        eq_count += 1
    return eq_count
```

2 Reproductions With the Given Parameters

2.1 Task 1

```
S = Sequence(array=[0, 1, 2])
print(S.array)
```

Output:

```
[0, 1, 2]
```

The input parameter is correctly passed into the array variable.

2.2 Task 3

```
AS = Arithmetic(start=1, step=2)
AS(length=5)
```

Output:

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

The Arithmetic sequence is correctly generated.

2.3 Task 4

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

Output:

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

```
5
```

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

The length of the sequence is correctly outputted. The sequence output in an iterative manner is the same with the original sequence.

2.4 Task 5

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

Output:

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

```
8
```

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

The Geometric sequence is correctly generated. The related methods work as desired.

2.5 Task 6

```
AS = Arithmetic(start=1, step=2)
AS(length=5)
GS = Geometric(start=1, ratio=2)
GS(length=5)
print(AS == GS)
GS(length=8)
print(AS == GS)
```

Output:

```
[1, 3, 5, 7, 9]
[1, 2, 4, 8, 16]
1
[1, 2, 4, 8, 16, 32, 64, 128]
Traceback (most recent call last):
  File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 106, in <module>
    print(AS == GS)
  File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 24, in __eq__
    raise ValueError('Two arrays are not equal in length!')
ValueError: Two arrays are not equal in length!
```

When the lengths of the two sequences are the same, the number of equal corresponding elements is correctly counted; otherwise, the exception is thrown as expected.

3 Outputs With Input Parameters of My Choice

3.1 Task 1

```
S = Sequence(array=[3, 6, 9])
print(S.array)
```

Output:

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

When the input parameter is changed, the array variable is changed correspondingly.

3.2 Task 3

```
AS = Arithmetic(start=2, step=4)
AS(length=4)
```

Output:

```
[2, 6, 10, 14]
```

The Arithmetic sequence is correctly generated when parameters change.

3.3 Task 4

```
AS = Arithmetic(start=2, step=4)
AS(length=4)
print(len(AS))
print([n for n in AS])
```

Output:

```
[2, 6, 10, 14]
4
[2, 6, 10, 14]
```

The length of the sequence is correctly outputted when parameters change. The sequence output in an iterative manner is the same with the original sequence.

3.4 Task 5

```
GS = Geometric(start=2, ratio=3)
GS(length=6)
print(len(GS))
print([n for n in GS])
```

Output:

```
[2, 6, 18, 54, 162, 486]
6
[2, 6, 18, 54, 162, 486]
```

The Geometric sequence is correctly generated when parameters change. The related methods work as desired.

3.5 Task 6

```
AS = Arithmetic(start=2, step=4)
AS(length=4)
GS = Geometric(start=2, ratio=3)
GS(length=4)
print(AS == GS)
GS(length=6)
print(AS == GS)
```

Output:

```
[2, 6, 10, 14]
[2, 6, 18, 54]
2
[2, 6, 18, 54, 162, 486]
Traceback (most recent call last):
  File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 160, in <module>
    print(AS == GS)
  File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 23, in __eq__
    raise ValueError('Two arrays are not equal in length!')
ValueError: Two arrays are not equal in length!
```

When the lengths of the two sequences are the same, the number of equal corresponding elements is correctly counted (in this case, the first two elements are equal respectively); otherwise, the exception is thrown as expected.

4 Source Code

```

1  ## ECE 601 - HW 1
2  # Author: Wei Xu
3
4  ## Task 1
5  class Sequence(object):
6      def __init__(self, array):
7          self.array = array
8      ## Task 4
9      def __len__(self):
10         return len(self.array)
11     def __iter__(self):
12         self.index = -1 # Initialize the index
13         return self
14     def __next__(self):
15         self.index += 1
16         if self.index < len(self):
17             return self.array[self.index]
18         else:
19             raise StopIteration
20     ## Task 6
21     def __eq__(self, other):
22         if len(self) != len(other):
23             raise ValueError('Two arrays are not equal in length!')
24         else:
25             eq_count = 0 # count the number of equal corresponding elements
26             for i in range(len(self)):
27                 if self.array[i] == other.array[i]:
28                     eq_count += 1
29             return eq_count
30
31     ## Task 2
32     class Arithmetic(Sequence):
33         def __init__(self, start, step):
34             super().__init__([])
35             self.start = start
36             self.step = step
37         ## Task 3
38         def __call__(self, length):
39             temp = self.start
40             self.array = [] # Reset the array
41             for _ in range(length):
42                 self.array.append(temp)
43                 temp += self.step # Calculate the next element
44             print(self.array) # Print after generating
45
46     ## Task 5
47     class Geometric(Sequence):
48         def __init__(self, start, ratio):
49             super().__init__([])
50             self.start = start
51             self.step = ratio
52         def __call__(self, length):
53             temp = self.start
54             self.array = [] # Reset the array

```

```

55         for _ in range(length):
56             self.array.append(temp)
57             temp *= self.step # Calculate the next element
58             print(self.array) # Print after generating
59
60 REPRODUCTION = True
61
62 if REPRODUCTION:
63
64     print('\n### Reproductions with the given parameters ###')
65
66     print('\n--- Task 1 ---')
67     S = Sequence(array=[0, 1, 2])
68     print(S.array)
69     ## Print:
70     # [0, 1, 2]
71
72     print('\n--- Task 3 ---')
73     AS = Arithmetic(start=1, step=2)
74     AS(length=5)
75     ## Print:
76     # [1, 3, 5, 7, 9]
77
78     print('\n--- Task 4 ---')
79     AS = Arithmetic(start=1, step=2)
80     AS(length=5)
81     print(len(AS))
82     print([n for n in AS])
83     ## Print:
84     # [1, 3, 5, 7, 9]
85     # 5
86     # [1, 3, 5, 7, 9]
87
88     print('\n--- Task 5 ---')
89     GS = Geometric(start=1, ratio=2)
90     GS(length=8)
91     print(len(GS))
92     print([n for n in GS])
93     ## Print:
94     # [1, 2, 4, 8, 16, 32, 64, 128]
95     # 8
96     # [1, 2, 4, 8, 16, 32, 64, 128]
97
98     print('\n--- Task 6 ---')
99     AS = Arithmetic(start=1, step=2)
100    AS(length=5)
101    GS = Geometric(start=1, ratio=2)
102    GS(length=5)
103    print(AS == GS)
104    GS(length=8)
105    print(AS == GS)
106    ## Print:
107    # [1, 3, 5, 7, 9]
108    # [1, 2, 4, 8, 16]

```

```

109 # 1
110 # [1, 2, 4, 8, 16, 32, 64, 128]
111 # Traceback (most recent call last):
112 #   File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 80, in <module>
113 #     print(AS == GS)
114 #   File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 23, in __eq__
115 #     raise ValueError('Two arrays are not equal in length!')
116 # ValueError: Two arrays are not equal in length!
117
118 else:
119
120     print('\n### Outputs with input parameters of my choice ###')
121     print('\n--- Task 1 ---')
122     S = Sequence(array=[3, 6, 9])
123     print(S.array)
124     ## Print:
125     # [3, 6, 9]
126
127     print('\n--- Task 3 ---')
128     AS = Arithmetic(start=2, step=4)
129     AS(length=4)
130     ## Print:
131     # [2, 6, 10, 14]
132
133     print('\n--- Task 4 ---')
134     AS = Arithmetic(start=2, step=4)
135     AS(length=4)
136     print(len(AS))
137     print([n for n in AS])
138     ## Print:
139     # [2, 6, 10, 14]
140     # 4
141     # [2, 6, 10, 14]
142
143     print('\n--- Task 5 ---')
144     GS = Geometric(start=2, ratio=3)
145     GS(length=6)
146     print(len(GS))
147     print([n for n in GS])
148     ## Print:
149     # [2, 6, 18, 54, 162, 486]
150     # 6
151     # [2, 6, 18, 54, 162, 486]
152
153     print('\n--- Task 6 ---')
154     AS = Arithmetic(start=2, step=4)
155     AS(length=4)
156     GS = Geometric(start=2, ratio=3)
157     GS(length=4)
158     print(AS == GS)
159     GS(length=6)
160     print(AS == GS)
161     ## Print:
162     # [2, 6, 10, 14]

```

```
163 | # [2, 6, 18, 54]
164 | # 2
165 | # [2, 6, 18, 54, 162, 486]
166 | # Traceback (most recent call last):
167 | #   File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 160, in <module>
168 | #     print(AS == GS)
169 | #   File "/home/weixu/Documents/ece601/hw1/hw1_WeiXu.py", line 23, in __eq__
170 | #     raise ValueError('Two arrays are not equal in length!')
171 | # ValueError: Two arrays are not equal in length!
```