Homework 1 • Graded

## Student

Sangwon Ji

**Total Points** 

2 / 2 pts

**Autograder Score** 

2.0 / 2.0

## **Autograder Results**

```
______
Assignment: Homework 1
OK, version v1.18.1
______
Scoring tests
Doctests for a_plus_abs_b
>>> from hw01 import *
>>> a_plus_abs_b(2, 3)
5
>>> a_plus_abs_b(2, -3)
5
>>> a_plus_abs_b(-1, 4)
>>> a_plus_abs_b(-1, -4)
3
Score: 1.0/1
Doctests for a_plus_abs_b_syntax_check
>>> from hw01 import *
>>> # You aren't expected to understand the code of this test.
>>> import inspect, re
>>> re.findall(r'^\s*(return .*)', inspect.getsource(a_plus_abs_b), re.M)
['return f(a, b)']
Score: 1.0/1
Doctests for two_of_three
>>> from hw01 import *
>>> two_of_three(1, 2, 3)
>>> two_of_three(5, 3, 1)
```

```
10
>>> two_of_three(10, 2, 8)
68
>>> two_of_three(5, 5, 5)
50
Score: 1.0/1
Doctests for two_of_three_syntax_check
>>> from hw01 import *
>>> # You aren't expected to understand the code of this test.
>>> import inspect, ast
>>> [type(x).__name__ for x in ast.parse(inspect.getsource(two_of_three)).body[0].body]
['Expr', 'Return']
Score: 1.0/1
Doctests for largest_factor
>>> from hw01 import *
>>> largest_factor(15) # factors are 1, 3, 5
5
>>> largest_factor(80) # factors are 1, 2, 4, 5, 8, 10, 16, 20, 40
40
>>> largest_factor(13) # factor is 1 since 13 is prime
1
Score: 1.0/1
Doctests for hailstone
>>> from hw01 import *
>>> a = hailstone(10)
10
5
16
8
4
2
1
>>> a
7
>>> b = hailstone(1)
1
>>> b
1
Score: 1.0/1
Point breakdown
a_plus_abs_b: 1.0/1
a_plus_abs_b_syntax_check: 1.0/1
two_of_three: 1.0/1
```

two\_of\_three\_syntax\_check: 1.0/1

largest\_factor: 1.0/1 hailstone: 1.0/1

Score: Total: 6.0

Cannot backup when running ok with --local.

Final Score:2.0

## **Submitted Files**

→ hw01.py **L** Download

```
1
     from operator import add, sub
2
3
4
     def a_plus_abs_b(a, b):
5
       """Return a+abs(b), but without calling abs.
6
7
       >>> a_plus_abs_b(2, 3)
8
9
       >>> a_plus_abs_b(2, -3)
10
       5
11
       >>> a_plus_abs_b(-1, 4)
12
       >>> a_plus_abs_b(-1, -4)
13
14
       3
       111111
15
16
       if b < 0:
17
         f = sub
18
       else:
19
         f = add
20
       return f(a, b)
21
22
23
     def a_plus_abs_b_syntax_check():
24
       """Check that you didn't change the return statement of a_plus_abs_b.
25
26
       >>> # You aren't expected to understand the code of this test.
27
       >>> import inspect, re
28
       >>> re.findall(r'^\s*(return .*)', inspect.getsource(a_plus_abs_b), re.M)
29
       ['return f(a, b)']
30
31
       # You don't need to edit this function. It's just here to check your work.
32
33
     def two_of_three(i, j, k):
34
       """Return m*m + n*n, where m and n are the two smallest members of the
35
36
       positive numbers i, j, and k.
37
38
       >>> two_of_three(1, 2, 3)
39
40
       >>> two_of_three(5, 3, 1)
41
       10
42
       >>> two_of_three(10, 2, 8)
43
       68
44
       >>> two_of_three(5, 5, 5)
45
       50
46
47
       return i*i + j*j + k*k - max(i,j,k) * max(i,j,k)
48
49
```

```
50
     def two_of_three_syntax_check():
        """Check that your two_of_three code consists of nothing but a return statement.
51
52
        >>> # You aren't expected to understand the code of this test.
53
54
        >>> import inspect, ast
55
        >>> [type(x).__name__ for x in ast.parse(inspect.getsource(two_of_three)).body[0].body]
        ['Expr', 'Return']
56
57
58
        # You don't need to edit this function. It's just here to check your work.
59
60
     def largest_factor(n):
61
        """Return the largest factor of n that is smaller than n.
62
63
64
        >>> largest_factor(15) # factors are 1, 3, 5
65
66
        >>> largest_factor(80) # factors are 1, 2, 4, 5, 8, 10, 16, 20, 40
67
        40
68
        >>> largest_factor(13) # factor is 1 since 13 is prime
69
        1
        111111
70
71
        i = n
72
        while n > 1:
73
          i = i - 1
74
          if n % i == 0:
75
             return i
76
77
78
     def hailstone(n):
79
        """Print the hailstone sequence starting at n and return its
80
        length.
81
82
        >>> a = hailstone(10)
83
        10
        5
84
85
        16
86
        8
        4
87
        2
88
89
        1
        >>> a
90
        7
91
92
        >>> b = hailstone(1)
93
        1
        >>> b
94
95
        1
        111111
96
97
        length1 = 0
98
99
        while n !=1:
100
          print(n)
          if n \% 2 ==0: # If n is even
101
```

```
102
           n = n//2
103
         else: # If n is odd
104
           n = 3 * n + 1
105
         length1 += 1
106
       print(n)
107
       length1 += 1
108
109
       return(length1)
110
111
```