hw01

November 16, 2021

[1]:
```
#Q2: A plus Abs B
from operator import add, sub

def a_plus_abs_b(a, b):
    """Return a+abs(b), but without calling abs.

    >>> a_plus_abs_b(2, 3)
    5
    >>> a_plus_abs_b(2, -3)
    5
    >>> # a check that you didn't change the return statement!
    >>> import inspect, re
    >>> re.findall(r'^\s*(return .*)', inspect.getsource(a_plus_abs_b), re.M)
    ['return f(a, b)']
    """
    if b < 0:
        f = sub
    else:
        f = add
    return f(a, b)
```

[2]:
```
a_plus_abs_b(2, 3)
```

[2]: 5

[3]:
```
a_plus_abs_b(2, -3)
```

[3]: 5

[4]:
```
#Q3: Two of Three
def two_of_three(x, y, z):
    """Return a*a + b*b, where a and b are the two smallest members of the
    positive numbers x, y, and z.

    >>> two_of_three(1, 2, 3)
    5
    >>> two_of_three(5, 3, 1)
```

```
    10
    >>> two_of_three(10, 2, 8)
    68
    >>> two_of_three(5, 5, 5)
    50
    >>> # check that your code consists of nothing but an expression (this␣
↪docstring)
    >>> # a return statement
    >>> import inspect, ast
    >>> [type(x).__name__ for x in ast.parse(inspect.getsource(two_of_three)).
↪body[0].body]
    ['Expr', 'Return']
    """
    return x**2 + y**2 + z**2 - max(x, y, z)**2
```

[5]:
```
two_of_three(1, 2, 3)
```

[5]: 5

[6]:
```
two_of_three(5, 3, 1)
```

[6]: 10

[7]:
```
two_of_three(10, 2, 8)
```

[7]: 68

[8]:
```
two_of_three(5, 5, 5)
```

[8]: 50

[9]:
```
#Q4: Largest Factor
from math import sqrt
def largest_factor(n):
    """Return the largest factor of n that is smaller than n.

    >>> 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
    """
    "*** YOUR CODE HERE ***"
    s = int(n/2)
    while s > 0:
        if n % s == 0:
```

```
            break
        else:
            s -= 1

    return s
```

[10]:
```
largest_factor(15)
```

[10]: 5

[11]:
```
largest_factor(80)
```

[11]: 40

[12]:
```
largest_factor(13)
```

[12]: 1

[13]:
```
#Q5: If Function vs Statement
def if_function(condition, true_result, false_result):
    """Return true_result if condition is a true value, and
    false_result otherwise.

    >>> if_function(True, 2, 3)
    2
    >>> if_function(False, 2, 3)
    3
    >>> if_function(3==2, 3+2, 3-2)
    1
    >>> if_function(3>2, 3+2, 3-2)
    5
    """
    if condition:
        return true_result
    else:
        return false_result

def with_if_statement():
    """
    >>> result = with_if_statement()
    47
    >>> print(result)
    None
    """
    if cond():
        return true_func()
    else:
```

```
        return false_func()

def with_if_function():
    """
    >>> result = with_if_function()
    42
    47
    >>> print(result)
    None
    """
    return if_function(cond(), true_func(), false_func())
"""
write functions cond, true_func, and false_func such that with_if_statement
prints the number 47, but with_if_function prints both 42 and 47.
"""
def cond():
    "*** YOUR CODE HERE ***"
    return False

def true_func():
    "*** YOUR CODE HERE ***"
    print(42)

def false_func():
    "*** YOUR CODE HERE ***"
    print(47)
```

[14]:
```
result = with_if_function()
```

```
42
47
```

[15]:
```
result = with_if_statement()
```

```
47
```

[16]:
```
#Q6: Hailstone
def hailstone(n):
    """Print the hailstone sequence starting at n and return its
    length.

    >>> a = hailstone(10)
    10
    5
    16
    8
```

```
    4
    2
    1
    >>> a
    7
    """
    "*** YOUR CODE HERE ***"
    step = 0
    while n != 1:
        print(n)
        if n % 2 == 0:
            n = n / 2
        else:
            n = 3 * n + 1
        step = step + 1
    print(n)
    step = step + 1
    return step
```

[17]:
```
a = hailstone(10)
```

```
10
5.0
16.0
8.0
4.0
2.0
1.0
```

[18]:
```
a
```

[18]: 7

[19]:
```
a = hailstone(27)
a
```

```
27
82
41.0
124.0
62.0
31.0
94.0
47.0
142.0
71.0
214.0
107.0
```

322.0
161.0
484.0
242.0
121.0
364.0
182.0
91.0
274.0
137.0
412.0
206.0
103.0
310.0
155.0
466.0
233.0
700.0
350.0
175.0
526.0
263.0
790.0
395.0
1186.0
593.0
1780.0
890.0
445.0
1336.0
668.0
334.0
167.0
502.0
251.0
754.0
377.0
1132.0
566.0
283.0
850.0
425.0
1276.0
638.0
319.0
958.0
479.0
1438.0

719.0
2158.0
1079.0
3238.0
1619.0
4858.0
2429.0
7288.0
3644.0
1822.0
911.0
2734.0
1367.0
4102.0
2051.0
6154.0
3077.0
9232.0
4616.0
2308.0
1154.0
577.0
1732.0
866.0
433.0
1300.0
650.0
325.0
976.0
488.0
244.0
122.0
61.0
184.0
92.0
46.0
23.0
70.0
35.0
106.0
53.0
160.0
80.0
40.0
20.0
10.0
5.0
16.0

```
8.0
4.0
2.0
1.0
```

[19]: 112