Homework 1 Solutions **hw01.zip (hw01.zip)**

Solution Files

You can find the solutions in hw01.py (hw01.py).

Required Questions

Q1: A Plus Abs B

Python's `operator` module contains two-argument functions such as `add` and `sub` for Python's built-in arithmetic operators. For example, `add(2, 3)` evalutes to 5, just like the expression `2 + 3`.

Fill in the blanks in the following function for adding `a` to the absolute value of `b`, without calling `abs`. You may **not** modify any of the provided code other than the two blanks.

```
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_plus_abs_b(-1, 4)
    3
    >>> a_plus_abs_b(-1, -4)
    3
    """
    if b < 0:
        f = sub
    else:
        f = add
    return f(a, b)
```

Use Ok to test your code:

```
python3 ok -q a_plus_abs_b
```

Use Ok to run the local syntax checker (which checks that you didn't modify any of the provided code other than the two blanks):

```
python3 ok -q a_plus_abs_b_syntax_check
```

If `b` is positive, we add the numbers together. If `b` is negative, we subtract the numbers. Therefore, we choose the operator `add` or `sub` based on the sign of `b`.

Q2: Two of Three

Write a function that takes three *positive* numbers as arguments and returns the sum of the squares of the two smallest numbers. **Use only a single line for the body of the function.**

```
def two_of_three(i, j, k):
    """Return m*m + n*n, where m and n are the two smallest members of the
    positive numbers i, j, and k.

    >>> 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
    """
    return min(i*i+j*j, i*i+k*k, j*j+k*k)
    # Alternate solution
def two_of_three_alternate(i, j, k):
    return i**2 + j**2 + k**2 - max(i, j, k)**2
```

Hint: Consider using the `max` or `min` function:

```
>>> max(1, 2, 3)
3
>>> min(-1, -2, -3)
-3
```

Use Ok to test your code:

```
python3 ok -q two_of_three
```

Use Ok to run the local syntax checker (which checks that you used only a single line for the body of the function):

```
python3 ok -q two_of_three_syntax_check
```

We use the fact that if `x>y` and `y>0`, then `square(x)>square(y)`. So, we can take the `min` of the sum of squares of all pairs. The `min` function can take an arbitrary number of arguments.

Alternatively, we can do the sum of squares of all the numbers. Then we pick the largest value, and subtract the square of that.

Q3: Largest Factor

Write a function that takes an integer `n` that is **greater than 1** and returns the largest integer that is smaller than `n` and evenly divides `n`.

```
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
    """
    factor = n - 1
    while factor > 0:
        if n % factor == 0:
            return factor
        factor -= 1
```

> **Hint:** To check if `b` evenly divides `a`, use the expression `a % b == 0`, which can be read as, "the remainder when dividing `a` by `b` is 0."

Use Ok to test your code:

```
python3 ok -q largest_factor
```

Iterating from `n-1` to 1, we return the first integer that evenly divides `n`. This is guaranteed to be the largest factor of `n`.

Q4: Hailstone

Douglas Hofstadter's Pulitzer-prize-winning book, *Gödel, Escher, Bach*, poses the following mathematical puzzle.

1. Pick a positive integer `n` as the start.
2. If `n` is even, divide it by 2.
3. If `n` is odd, multiply it by 3 and add 1.
4. Continue this process until `n` is 1.

The number `n` will travel up and down but eventually end at 1 (at least for all numbers that have ever been tried -- nobody has ever proved that the sequence will terminate). Analogously, a hailstone travels up and down in the atmosphere before eventually landing on

earth.

This sequence of values of `n` is often called a Hailstone sequence. Write a function that takes a single argument with formal parameter name `n`, prints out the hailstone sequence starting at `n`, and returns the number of steps in the sequence:

```
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
    >>> b = hailstone(1)
    1
    >>> b
    1
    """
    length = 1
    while n != 1:
        print(n)
        if n % 2 == 0:
            n = n // 2      # Integer division prevents "1.0" output
        else:
            n = 3 * n + 1
        length = length + 1
    print(n)                # n is now 1
    return length
```

Hailstone sequences can get quite long! Try 27. What's the longest you can find?

> Note that if `n == 1` initially, then the sequence is one step long.
> **Hint:** If you see 4.0 but want just 4, try using floor division `//` instead of regular division `/`.

Use Ok to test your code:

```
python3 ok -q hailstone
```

Curious about hailstone sequences? Take a look at this article:

- In 2019, there was a major development (https://www.quantamagazine.org/mathematician-terence-tao-and-the-collatz-conjecture-20191211/) in understanding how the hailstone conjecture works for most numbers!

We keep track of the current length of the hailstone sequence and the current value of the hailstone sequence. From there, we loop until we hit the end of the sequence, updating the length in each step.

> Note: we need to do floor division `//` to remove decimals.

Check Your Score Locally

You can locally check your score on each question of this assignment by running

```
python3 ok --score
```

This does NOT submit the assignment! When you are satisfied with your score, submit the assignment to Gradescope to receive credit for it.

Submit Assignment

Submit this assignment by uploading any files you've edited **to the appropriate Gradescope assignment.** Lab 00 (../../lab/lab00/#submit-with-gradescope) has detailed instructions.