CS 61A: Quiz 1

Due by 11:59pm on Thursday, 1/29

Instructions

Download quiz01.zip. Inside the archive, you will find a file called quiz01.py, along with a copy of the OK autograder.

Complete the quiz and submit it before 11:59pm on Thursday, 1/29. **You must work alone**, but you may talk to the course staff (see **Asking Questions** below). You may use any course materials, including an interpreter, course videos, slides, and readings. Please **do not** discuss these specific questions with your classmates, and **do not** scour the web for answers or post your answers online.

Your submission will be graded automatically for correctness. Your implementations **do not** need to be efficient, as long as they are correct. We will apply additional correctness tests as well as the ones provided. Passing these tests does not guarantee a perfect score.

Asking Questions: If you believe you need clarification on a question, **make a private post** on Piazza. Please do not post publicly about the quiz contents. If the staff discovers a problem with the quiz or needs to clarify a question, we will email the class via Piazza. You can also come to office hours to ask questions about the quiz or any other course material, but no answers or hints will be provided in office hours.

Submission: When you are done, submit with `python3 ok --submit`. You may submit more than once before the deadline; only the final submission will be scored.

Using OK

The `ok` program helps you test your code and track your progress. The first time you run the autograder, you will be asked to log in with your @berkeley.edu account using your web browser. Please do so. Each time you run ok, it will back up your work and

progress on our servers. You can run all the doctests with the following command:

```
python3 ok
```

To test a specific question, use the `-q` option with the name of the function:

```
python3 ok -q <function>
```

By default, only tests that **fail** will appear. If you want to see how you did on all tests, you can use the `-v` option:

```
python3 ok -v
```

If you do not want to send your progress to our server or you have any problems logging in, add the `--local` flag to block all communication:

```
python3 ok --local
```

When you are ready to submit, run `ok` with the `--submit` option:

```
python3 ok --submit
```

Readings: You might find the following references useful:

- Section 1.2
- Section 1.3
- Section 1.4
- Section 1.5

You can watch a video of the solution to Fall 2014 quiz 1 if you want to see an example

of how to solve similar problems.

Table of Contents

Question 1

Implement `harmonic`, which returns the harmonic mean of two positive numbers `x` and `y`. The harmonic mean of `2` numbers is `2` divided by the sum of the reciprocals of the numbers. (The reciprocal of `x` is `1/x`.)

```
def harmonic(x, y):
    """Return the harmonic mean of x and y.

    >>> harmonic(2, 6)
    3.0
    >>> harmonic(1, 1)
    1.0
    >>> harmonic(2.5, 7.5)
    3.75


    """
    "*** YOUR CODE HERE ***"
```

Test your code using OK:

```
python3 ok -q harmonic
```

Question 2

Complete the implementation of `pi_fraction`, which takes a positive number `gap` and prints the fraction that is no more than `gap` away from `pi` and has the smallest

possible positive integer denominator. See the doctests for the format of the printed output.

Hint: If you want to find the nearest integer to a number, use the built-in `round` function. It's possible to solve this problem without using `round`.

You may change the starter implementation if you wish.

```
from math import pi

def pi_fraction(gap):
    """Print the fraction within gap of pi that has the smallest de

    >>> pi_fraction(0.01)
    22 / 7 = 3.142857142857143
    >>> pi_fraction(1)
    3 / 1 = 3.0
    >>> pi_fraction(1/8)
    13 / 4 = 3.25
    >>> pi_fraction(1e-6)
    355 / 113 = 3.1415929203539825


    """
    numerator, denominator = 3, 1
    "*** YOUR CODE HERE ***"
    print(numerator, '/', denominator, '=', numerator/denominator)
```

Test your code using OK:

```
python3 ok -q pi_fraction
```

Question 3

Implement the function `nearest_two`, which takes as input a positive number `x` and returns the power of two (..., 1/8, 1/4, 1/2, 1, 2, 4, 8, ...) that is nearest to `x`. If `x` is exactly between two powers of two, return the larger.

You may change the starter implementation if you wish.

```
def nearest_two(x):
    """Return the power of two that is nearest to x.

    >>> nearest_two(8)    # 2 * 2 * 2 is 8
    8.0
    >>> nearest_two(11.5) # 11.5 is closer to 8 than 16
    8.0
    >>> nearest_two(14)   # 14 is closer to 16 than 8
    16.0
    >>> nearest_two(2015)
    2048.0
    >>> nearest_two(.1)
    0.125
    >>> nearest_two(0.75) # Tie between 1/2 and 1
    1.0
    >>> nearest_two(1.5)  # Tie between 1 and 2
    2.0


    """
    power_of_two = 1.0
    "*** YOUR CODE HERE ***"
    return power_of_two
```

Test your code using OK:

```
python3 ok -q nearest_two
```