# Com S 127x
# Fall 2016
# Assignment 2

Due Date: Wednesday, September 21, 11:59 pm (midnight)

"Late" deadline (25% penalty): Thursday, September 22, 11:59 pm

## General information

**This assignment is to be done on your own. See the Academic Dishonesty policy in the syllabus for details.**

If you need help, see your instructor or one of the TAs. Lots of help is also available through the Piazza discussions.

Please start the assignment as soon as possible and get your questions answered right away.

## Problem 1

Write a Python program `rps.py` to play a game of *rock, paper, scissors* against you. More specifically, your code should:

1. prompt the user to enter "r", "s", or "p"
2. generate one of the three values "r", "s", or "p" at random
3. print the outcome of the game as one of the following strings:
   - `"tie"`
   - `"rock crushes scissors"`
   - `"paper covers rock"`
   - `"scissors cut paper"`

(We're using letters "r", "p" and "s" just to save typing.) If the user enters something other than "r", "p", or "s", you should just print "error".

(If you are not familiar with the game, see https://en.wikipedia.org/wiki/Rock–paper–scissors.)

To generate the random value, first generate a random number 0, 1, or 2. To do this, you'll need to import the module `random`. Then use the line:

```
num = random.randrange(0, 3)
```

That gives you a random value **num** of 0, 1, or 2. Once you have that, you can use a conditional statement to convert it to "r", "s", or "p". (Or, if you'd rather, you could convert the user's input to a number 0, 1, or 2 and work with the numbers.)

> (*Food for thought*: If you use the numbers 0, 1, and 2 to represent rock, scissors, and paper (in that order), can you determine who won the game using arithmetic, instead of using a complicated if-statement to check all the cases?)

## Problem 2

The U.S. Postal Service has some complex rules for deciding how much it costs to mail a letter. For this assignment we'll use a small part of these rules. Your task is to write an interactive Python program, **postage.py**, that inputs the weight of a letter in ounces, and prints the cost of mailing it.

For an ordinary letter up to 3.5 ounces, the cost is .47 plus .21 for each *additional* ounce (or *part* of an ounce)[1]. Try some examples:

- A letter that is 1 ounce (or less) is just .47.
- For a 3 ounce letter the cost is .47 + (2 * .21) = .89
- For a 2.3 ounce letter, the cost is also .47 + (2 * .21) = .89, since 1.3 rounds up to 2
- For a 3.1 ounce letter the cost is .47 + (3 * .21) = 1.10, since 2.1 rounds up to 3

To translate this into Python code, it would be nice to have a slick way to take a float value and round "up" to the next whole number. Fortunately there is a function in the **math** module, called **ceil()** (short for "ceiling") that does exactly what we want. Try this in the shell:

```
>>>  import math
>>>  math.ceil(1.3)
2.0
>>>  math.ceil(2.0)
2.0
>>>  math.ceil(2.1)
3.0
```

Now, to make things more interesting, let's add the rule for letters *over* 3.5 ounces: the cost is .94 plus .21 for each additional ounce (or part of an ounce).

[1] If you haven't bought stamps since last year, you might be surprised to notice that US postage rates have actually gone *down* for the first time in pretty much the history of the universe. (A first class stamp was 49 cents last year.)

For example,

- a 3.5 ounce letter would cost .47 + (3 * .21) = 1.10, using the first rule
- a 3.8 ounce letter would cost .94 + (3 * .21) = 1.57
- a 10 ounce letter would cost .94 + (9 * .21) = 2.83

Notice you have to check the *actual* weight to see if it is bigger than 3.5, but then *round up* when you calculate the extra ounces.

*We are not requiring you to turn a pencil-and-paper worksheet, but you will find it much easier to write the code if you first work out some examples (such as the ones above) by hand. Your code can usually follow the same logic.*

*Be sure to carefully* ***test*** *your code on known sample values!*

## If you have questions

For questions, please see the Piazza Q & A pages and click on the folder **`hw1`**. If you don't find your question answered, then create a new post with your question. Try to state the question or topic clearly in the title of your post, and attach the tag **`hw1`**. *But remember, do not post any source code for the classes that are to be turned in.* It is fine to post source code for general Python examples that are not being turned in. (In the Piazza editor, use the button labeled "pre" to have code formatted the way you typed it.)

If you have a question that absolutely cannot be asked without showing part of your source code, make the post "private" so that only the instructors and TAs can see it.

## Documentation and style

Include comments at the top of each file with

- your name, and
- a brief statement of what the program does

## What to turn in

Upload your two files **`rps.py`** and **`postage.py`** to the Homework 2 submission link on Blackboard. Be sure to CHECK your submission history and make sure that you successfully submitted both files. **Remember you can't just upload the files, you have to click the "Submit" button!**