Homework 5

Decorators and Regular Expressions

Given February 19th – Due March 3rd

Exercise 1. Slow down decorator.
Write a @slowDown decorator that will sleep a certain amount of seconds before it calls the decorated function. Use an optional rate argument that controls how long it sleeps. Use a default value of 1 second.

Exercise 2. Fibonacci sequence
The Fibonacci Sequence is the series of numbers:

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
```

The next number is found by adding up the two numbers before it.

- The 2 is found by adding the two numbers before it (1+1)
- The 3 is found by adding the two numbers before it (1+2),
- The 5 is (2+3),
- and so on!

An implementation of the Fibonacci function could be as follows:

```
def fibonacci(num):
    if num < 2:
        return num
    return fibonacci(num - 1) + fibonacci(num - 2)
```

But the runtime performance is terrible. This is because the code keeps recalculating Fibonacci numbers that are already known.

1. Create a @cache decorator that will save the calculations in a function attribute dictionary. Make the decorator work for functions with more than one argument.
2. Compare with the @countCalls decorator the difference between using @cache and not using it. Write your conclusion at the top multiline comment under the “Description of your program”.

Exercise 3. Phone Number and Email Address Extractor

Say you have the boring task of finding every phone number and email address in a long web page or document. If you manually scroll through the page, you might end up searching for a long time. But if you had a program that could search the text in your clipboard for phone numbers and email addresses, you could simply press ctrl-A to select all the text, press ctrl-C to copy it to the clipboard, and then run your program. It could replace the text on the clipboard with just the phone numbers and email addresses it finds.

So, your phone and email address extractor will need to do the following:

- Get the text off the clipboard.
- Find all phone numbers and email addresses in the text.
- Paste them onto the clipboard.

Exercise 4. Strong Password Detection

Write a function that uses regular expressions to make sure the password string it is passed is strong. A strong password is defined as one that is at least eight characters long, contains both uppercase and lowercase characters, and has at least one digit. You may need to test the string against multiple regex patterns to validate its strength.

Submit your code files as *hm5_name_surname_ex_num.py*, where *num* is the exercise number 1, 2, etc. Comment everything so we know you wrote the code! On top of your files write this multiline comment with your information:

”””

Homework 5, Exercise 1 (or 2...)
Name
Date
Description of your program.

”””