Name: _____

PIC 97, Winter 2016 – Midterm Exam 1

Administered 7:00 p.m. – 9:00 p.m. Wednesday, January 27, 2016. Corresponds with 1W – 4T.
No outside resources other than features of Python software (Spyder, IPython, etc…), official Python documentation (provided by python.org), and the provided links (in the exam assignment page on CCLE) are allowed. No other websites (including those linked from the allowed websites) are acceptable.

Flip this sheet only when instructed.

(Until then, squint as much as you wish trying to read backwards through the page.)

The exam questions are contained in this document, but you are to answer in a provided Jupyter notebook template (available for download from CCLE). When in doubt, include a print statement that shows the results of your calculations. For instance, if the program asks you to produce a list, print it, too. When you are done with the exam, submit your completed Jupyter notebook on CCLE. You are welcome to write code in Spyder or other software installed on the PIC machines and copy it into the Jupyter notebook when you're done.

If you can't do a problem, or you think it will take too long, consider starting with the easy stuff, saving and uploading, then working on the harder things as time permits.

For now, log on to your computer. You may also download the exam materials from Week 4 of CCLE and get familiar with the allowed resources.

There are two problems marked as "Optional" on the exam. If you include code for one of these problems in the corresponding cell of your Jupyter notebook, it will be considered an attempt at the problem, and the attempt will be graded with the rest of the exam. If you enter *only* `# not attempted` in that cell, it will not count for or against your grade.

One problem has been marked as "Eliminated", but space is left in the template Jupyter notebook to preserve formatting. You can ignore this problem entirely.

The references linked from CCLE are:

http://www.tutorialspoint.com/python/python_exceptions.htm
https://www.jeffknupp.com/blog/2013/02/06/write-cleaner-python-use-exceptions/
http://www.tutorialspoint.com/python/python_classes_objects.htm
http://www.rafekettler.com/magicmethods.html
http://anandology.com/python-practice-book/iterators.html
http://www.tutorialspoint.com/python/python_files_io.htm
http://www.pythonforbeginners.com/systems-programming/using-the-csv-module-in-python/
http://regexone.com/references/python
http://pythex.org/

Good luck!

Name: _____

1. (Optional) You have a great bank that offers 10% interest every 6 months. Clearly your money will grow very fast, so you need to use Python as a calculator to determine how much you'll have in ten years. The formula for interest calculations is:
$$A = P\left(1 + \frac{r}{n}\right)^{nt}$$
Where *P* is the principal ($2000), *r* is the interest rate (10%), *n* is the number of times per year the interest is compounded (2 times per year), and *t* is the time the interest is allowed to compound in years (10 years). Complete the cell in the provided notebook accordingly, and print the result of this calculation.
2. (Eliminated. You don't need to do anything in the corresponding cell of the spreadsheet.)
3. Use a `while` loop to determine the minimum number of bits required to store the magnitude 23490134. Equivalently, use a `while` loop to determine how many digits are in the binary representation of the decimal 23490134. You may not use `math.log` or any function/method for converting decimals to another base; you are to use basic features of the Python language, and in particular, demonstrate the ability to use a `while` loop.
If you are unfamiliar with bits and binary, all you need to know is that there are two possibilities for a single bit - 0 or 1, which makes it possible to represent the *two* decimal values 0 or 1. With an additional bit, the number of possibilities doubles - 00, 01, 10, 11 – representing the *four* values 0, 1, 2, 3; and so on. You do not need to determine the actual binary representation; just the number of binary digits/bits required to represent it.
4. (Optional) Write a call to a single Python built-in function such that it returns a list of the odd numbers from 9587 to 9551, *inclusive*. (You are expected to know the appropriate function.)
5. Write a lambda expression for the probability density function of the normal distribution:
$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
6. Use `filter` to create a new list with only the numerical (non-string) elements of:
`["one", 2, 1, 2, "three", "five", 5, 6, "seven", 8]`
The same code should work with any list of strings and integers.
7. Use a `for` loop to print out only the strings with four or more letters AND the even integers from the original list (above). The same code should work with any list of strings and integers.
8. Write a program that asks the user to "`Enter positive integer n:`".
If the input is valid, use list comprehension to produce a list of `n` positive multiples of 3, beginning with `3` (i.e. `[3, 6, 9, …]`).
Otherwise, print "`That was not a positive integer.`"
Your program is expected to behave properly for any input, yet you should not (need to) write an exhaustive list of conditional statements. (About two conditions is OK if you need them).

Name: _____

9. Use Python to determine the number of unique words in `words.txt` *without* using a dictionary. You may use other built-in data structures if you wish. All words are separated by spaces, and lines are terminated with the newline character. Hyphenated words like "`care-free`" are considered to be a single word.
10. The file `heights.csv` contains the heights of 100 people (in cm).
    a. Without using `pandas`, load the data (excluding the column names) into the built-in Python data structure that would be fastest for retrieving the height, given a name.
    b. From that data structure, create a generator (using a generator expression, if desired) to yield the heights.
    c. Using the resulting generator, calculate the average height (and print it).

    Comment your code so we can tell which objects satisfy parts a. and b.

11. Create a class `MyList` such that:

```
a. >>> x = MyList([10, 30, 40 , 50])
   >>> print x[1]       # MyList is 1-indexed
   10

b. >>> x[2] = 20;
   >>> print x          # Set element operation inserts
   [10, 20, 30, 40, 50]

c. >>> for el in x:     # MyList is iterable
   >>>     print el,
   10 20 30 40 50
```

    Yes, you may-use Python built-in lists in the implementation of your `MyList` class. Yes, your class should work similarly with any list provided as an argument to the constructor. No, it is not correct if it prints `50` before `10 20 30 40 50`. (This is expected to be a common error).
12. Write a python command that (separately) captures the subdomains, domain names, and top level domains from the text in `urls.txt`. Your results should be a list of tuples, like:
    `[('keep', 'google', 'com'), ('', 'pythex', 'org'), ('', 'gpc', 'edu'), ('', 'gpc', 'edu'), ('giving', 'gpc', 'edu'), ('giving', 'gpc', 'edu')]`
    where "keep" is a subdomain, "google" is a domain, and "com" is a top level domain. Note that subdomains may or may not be present, and “www” is not captured.