Final Revision: 1/24/2014

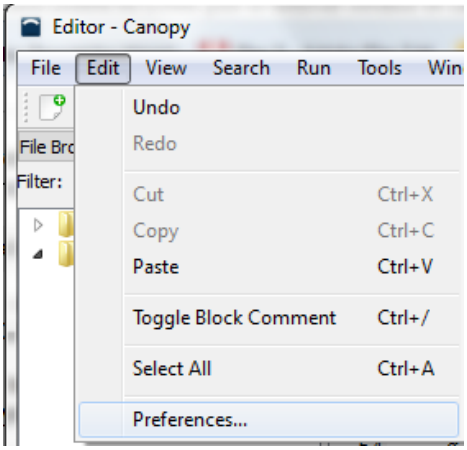
HW2: CS 110X C 2014

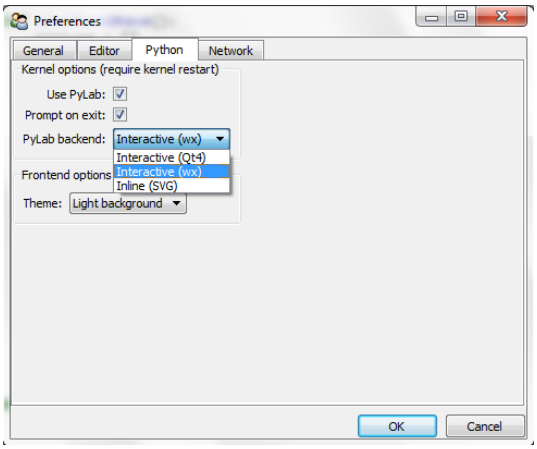
Note: This homework (and all remaining homework assignments) is a **partner homework** and must be completed by each partner pair. When you complete this assignment, you must not share your answers with any other student. Only one person from a partner pair needs to submit the assignment, but make sure that you submit before the deadline!

Note: I have revised this homework to slow down the pace of the course. Some questions removed from this assignment will appear in HW3.

Canopy Issues

If you are running Canopy then you have to make a small configuration change for this homework to work properly. From within the Canopy Editor, select menu item **Edit | Preferences…**.





Then in the Preferences window, select the **Python** tab and be sure that **PyLab backend** is set to "Interactive (wx)". I have tested this on Windows and on a MacBook.

Homework Instructions

This **revised** Homework has **eight** questions.

For each question be sure you understand exactly the format of the output that is requested. You will lose points if you do not exactly follow the format of the output for the individual questions. Should you have any questions, be sure to post on the HW2 discussion forum.

Q1	Demonstrate function returning values
Skills PM-1 PF-2 PF-3 Lecture Dependency *Jan-23*	Sound waves are constructed based on Sinusoidal data, which can easily be computed using the mathematical Sine function. For this problem define a function `sin(x)` that uses the Taylor series to approximate this value using six terms as follows: $$\sin(x) \cong x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!}$$ Note that $n! = n * (n-1) * (n-2) * \ldots * 3 * 2 * 1$ and x is a real number. Your function must **return** a value, not just print it out to the console. Note that because this computation is an approximation, there are some inputs which will result in values that are not within the expected [-1, 1] range of the Sine function.

Sample Output in IDLE	`>>> sin(3.1415)`
	`-0.00035233720521839814`
	`>>> sin(1.5708)`
	`0.9999999437325972`
	`>>> sin(7)`
	`-11.842203107463526`

Sample Output in Canopy	`In[3]: sin(3.1415)`
	`Out[3]: -0.00035233720521839814`
	
	`In[4]: sin(1.5708)`
	`Out[4]: 0.9999999437325972`

*Note: Canopy more clearly shows the return value of the sin() function by the **Out[n]** declaration which shows the value returned by the function that was invoked on the **In[n]** line. If a function has no **return** statement, then there is no **Out[n]** line in response to a function invocation.*

Q2	Demonstrate definite **for** loop
Skills PF-2 PF-3 CS-5 CS-9 SM-2 SM-3 PM-5 *Lecture Dependency* *Jan-23*	The Taylor expansion from Question Q1 is an infinite computation and you can compute as many terms as you wish. The `sin(x)` function you wrote in Q1 computes only six terms in the expansion, but a user may wish for more for increased accuracy. $$\sin(x) \cong x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \cdots$$ Define a function `taylorSin (x, n)` that computes n terms in the Taylor series. When invoked with n=6 the function result will be identical to the `sin(x)` function you wrote for Question Q1. To demonstrate your knowledge of the **for** loop, your function must **print** information as it approximates the function value. Follow the format below exactly to receive full credit. I am asking you to print this information with each pass because I believe it will help you understand the dynamic behavior of **for** loops. There must be exactly n lines of output, and each line shows the sign of the term being added, the exponent of the x value and the approximation computed so far, that is, after each pass. *You will find the math.factorial(n) function quite useful but you can still complete this problem without it.* Note that the signs alternate and the exponents are all increasing odd numbers.

Sample Output in IDLE	`>>> taylorSin(3.14,3)`
	`sign = 1, exponent= 1, approximation= 3.14`
	`sign = -1, exponent= 3, approximation= -2.01985733333`
	`sign = 1, exponent= 5, approximation= 0.523849134853`
	`0.5238491348533336`

Sample Output in Canopy	`In[1]: taylorSin(3.14, 3)`
	`sign = 1, exponent= 1, approximation= 3.14`
	`sign = -1, exponent= 3, approximation= -2.01985733333`
	`sign = 1, exponent= 5, approximation= 0.523849134853`
	`Out[1]: 0.5238491348533336`

Q3	Demonstrate knowledge of **if** statement, **else**, **elif**, definite **for** loop
Skills CS-1 CS-2 CS-3 CS-9 DT-10 Lecture Dependency Jan-24	Given a **list** of `'Yes'` and `'No'` string literal values, determine the result of a vote, namely: • A win for Yes (more Yes than No votes) • A win for No (more No than Yes votes) • A tie (same number of Yes and No votes) Define a function `tallyVote(votes)` that prints the results of the votes as recorded in a list that contains only `'Yes'` and `'No'` string literals.
Sample Output in IDLE	`>>> tallyVote(['Yes', 'No', 'Yes', 'No', 'No'])` `A win for No`
Sample Output in Canopy	`In[1]: tallyVote(['Yes', 'No', 'Yes', 'No'])` `A tie`

Q4	Incrementally construct a list
Skills PF-3 DT-6 *Lecture Dependency* *Jan-24*	In mathematics, an **arithmetic sequence** is a sequence of numbers such that the difference between the consecutive terms is constant. For instance, the sequence 5, 7, 9, 11, 13, 15 … is an arithmetic progression with common difference of 2 that starts at 5. A sequence is uniquely determined by a_0 (starting value), d (the common difference) and n (the number of terms to generate). You are to write a Python function that returns a list containing the values in such a sequence. Define a function `arithmeticSequence(a0, d, n)` that returns a list containing the first n terms in the sequence.
Sample Output in IDLE	`>>> arithmeticSequence(5, 2.5, 7)` `[5, 7.5, 10.0, 12.5, 15.0, 17.5, 20.0]`
Sample Output in Canopy	`In[1]: arithmeticSequence(1, 1, 10)` `Out[1]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`

Q5	Demonstrate knowledge of **if** statement, definite **for** loop
Skills CS-1 CS-2 CS-3 CS-9 DT-10 Lecture Dependency Jan-24	Given a **list** of values, return a new **list** that contains just the numbers that are within a designated range. Define a function `restrict(collection, low, high)` that returns a list containing the elements in collection that are ≥ low and ≤ high. Note that if none of the elements falls within the designated range then the empty list `[]` is returned.
Sample Output in IDLE	`>>> restrict([5, 3, 11, -5, 2, 7], 1, 7)` `[5, 3, 2, 7]`
Sample Output in Canopy	`In[1]: restrict ([5, 3, 11, -5, 2, 7], 1, 7)` `Out[1]: [5, 3, 2, 7]`

Q6	Incrementally construct a list
Skills PF-3 DT-6 *Lecture Dependency* *Jan-24*	When plotting real valued functions, you often need to create a list of evenly space x-coordinates drawn from some range [low, high]. Define a function `generateSamples(low, high, number)` that returns a new list containing the desired number of evenly spaced samples that includes both `low` and `high` in the list. Low and high can be integers or real numbers. You can assume that `low` will always be strictly smaller than high.
Sample Output in IDLE	`>>> generateSamples(1.5, 2.5, 5)` `[1.5, 1.75, 2.0, 2.25, 2.5]`
Sample Output in Canopy	`In[1]: generateSamples(1,5,4)` `Out[1]: [1.0, 2.333333333333333, 3.6666666666666665, 5.0]`

Q7	Demonstrate knowledge of **if** statement, definite **for** loop, function invocations

Skills
CS-1
CS-2
CS-3
CS-9
DT-10

Lecture Dependency
Jan-24

Write a function `computeStats(collection)` that allows the user to show a desired statistic for the given **list** passed into the function.

- Average value in the list
- Minimum value in the list
- Maximum value in the list
- Sum of the values in the list
- Number of values in the list in the range [1,5] (inclusive)

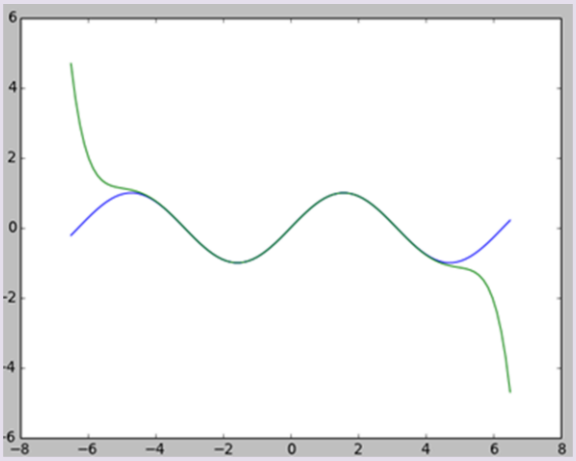
This function prompts the user for information as follows:

```
>>> computeStats([2,4,5,7,8])
Select desired statistic
  1. Average
  2. Minimum value
  3. Maximum value
  4. Sum
  5. Number of values in list in range [1,5]
Choose your statistic [1-5]: 3
Maximum = 8
```

The output produces the desired statistics as selected by the user. If the user doesn't choose a valid choice, simply print "Unknown request!"

Note that for option #5 you must use the `restrict()` function that you wrote for question Q5.

Be sure to output a useful bit of text with each computed value so the user can verify they see their desired statistic. For example, above you see "Maximum = 8" which is preferable to simply printing the value "8" on the line by itself.

Q8	Demonstrate ability to use pylab to plot values
Skills CS-1 CS-2 CS-3 CS-9 DT-10 Lecture Dependency Jan-27	Often you will need to generate a plot of a real-valued function. Fortunately, Python comes with a `pylab` module that will Define a function `plotRealSinVersusComputed(low, high, number)` that uses `pylab` to plot both functions in the given [low, high] range with a number of evenly spaced x-coordiantes. • The `sin(x)` function you wrote for question 1 • The `math.sin(x)` function provided by Python This function must use the `generateSamples()` function that you wrote for question Q6. The window that appears will contain a plot that looks like the following:  As you will see, the `sin(x)` function you wrote is a good approximation within a narrow range of x values.
Sample Output in IDLE	`>>> plotRealSinVersusComputed(-6.5, 6.5, 100)`

How To Get Started On This Assignment

A template HW2.py file is provided to you with some sample functions already provided.

Much of the work for this assignment will be spent trying to understand the domain of sound waves and writing the appropriate Python code. In many ways, that is as it should be! The job of a programmer is more than learning a particular syntax. You need to know how to produce code relevant for a specific problem. Sometimes the code you write is only 5 lines of code (but it will be just the right five lines of code).

Final Revision: 1/24/2014

Submit your HW2.py file using the web-based turnin system. As we have mentioned in class, only one of the team members needs to submit the assignment. But just make sure that something gets submitted!

Change Log

1. Moved some questions into HW3 and added a bunch more
2. Revised the sample output for Q2 which had `taylorSin(3.1415,3)` when it should have been `taylorSin(3.14,3)`
3. Clarified sample output for Q4