CS150 - Test Project 1
Due: Friday Oct. 21, at 6pm
(underneath my door if I'm not there)

A test project is an assignment that you complete on your own, without the help of others. It is a form of take-home exam. You may consult your text, your notes, your previous assignments, the notes and examples on the course web page and the Python library documentation (linked on the course web page), but use of any other source is forbidden. You may not discuss these problems with anyone aside from the course instructor. You may only ask the tutors for help with hardware problems or difficulties in retrieving your program.

Complete each of the three problems below, each as a separate program, in a separate file. You are encouraged to reuse code from your assignments or our class examples. Partial credit will be awarded, so if you can't solve a whole problem, get as far as you can.

Improving the programs and extra credit

If you do everything that is required, the maximum number of points you can get is 60. In order to get the full credit of 62, you must implement some extra features for some of the problems. I've provided some ideas below, but you should feel free to exercise your creativity. In addition, you may receive up to 3 points of additional extra credit (i.e. for a total of 65). You may receive at most 3 points of extra credit for any given program.

For all additions, include in the comments at the top of the file what you've added or you may not get credit.

1 Taxes

> *But in this world nothing can be said to be certain, except death and taxes.*
>
> –Benjamin Franklin

In preparation for entering the real-world, we're going to write a program that roughly estimates your federal taxes for the year. When you run the program, the following should happen:

- The user should be prompted for:

  - their name
  - their hourly wage
  - the number of hours they work a week
- The program will then print out how much the user makes a year
- and an estimate or how much they will owe in taxes

For example, here is a sample transcript from the program running:

```
Enter your name: Dave
Enter your hourly salary: 15
Enter your hours per week: 50
Dave, you make $42900.0 a year and will owe
$10725.0 in taxes
```

You **must** write the following functions:

- `salary_calculator`: Takes two parameters, hours worked in a week and the hourly wage and calculates the weekly wage. The first 40 hours a person works in the week are at the normal wage. Any hours above 40 that a person works are overtime and are billed at 1.5 times their normal wage. For example, if a person works 50 hours at $10/hour then they would make $550 ($400 regular and $150 of overtime).
- `tax_bracket`: Takes a single parameter, a *yearly* income, and returns the tax bracket as a the fraction of income would be required to be payed, e.g. 10% would be 0.1. You can find the tax brackets online at:

  `http://www.efile.com/tax-service/tax-calculator/tax-brackets/`

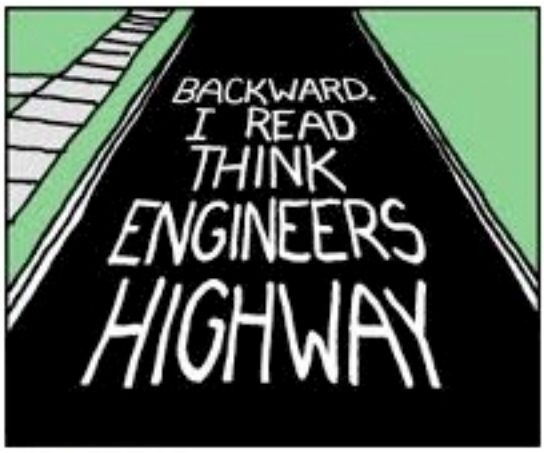
  We will use these as hard cutoffs, for example, if you made $30,000, `tax_bracket` would return 0.15[1]
- `calculate_taxes`: Takes as input a yearly salary and returns the amount of money a person with that yearly salary would owe in taxes (based on their tax bracket).

In addition to these functions, you may also include other functions and will need to include additional code to make your program behave appropriately.

Possible program extensions: provide information about monthly income after taxes, calculate the projected salary after some number of years assuming a fixed percent salary increase, allow the user to enter some tax write-offs that would decrease the yearly salary.

[1] In actuality, taxes are progressive, in that the first $8500 you make is taxed at 10%, then the amount from $8500-$34,500 at 15%, etc.

2 Word Play



http://xkcd.com/781/

Write a program that when you run it, prompts the user to enter a word and then prints out information about the word. For example, the following are the output from two different runs of the program:

```
Enter a word: banana
banana has 6 letters in it.
It is not a palindrome.
It starts with the letter 'b' and ends with 'a'
and has 3 vowels in it.
```

or another:

```
Enter a word: racecar
racecar has 7 letters in it.
It is a palindrome.
It starts with the letter 'r' and ends with 'r'
and has 3 vowels in it.
```

You must implement the following functions:

- `ispalindrome`: Takes a word as a parameter and returns a `bool` indicating whether or not the word is a palindrome. A palindrome is a word that is spelled the same way forwards and backwards (e.g. eye, anna, deleveled and testset). There are a number of ways to answer this question, so think about what would be the easiest to program.
- `vowel_count`: Takes a word as a parameter and returns the number of vowels that word has.

In addition to these functions, you may also include other functions and will need to include additional code to make your program behave appropriately.

Possible program extensions: add other characteristics about the word (e.g. number of consonants), allow the user to repeatedly enter words instead of just one (provide some way of exiting too)

3 Visualizing pi

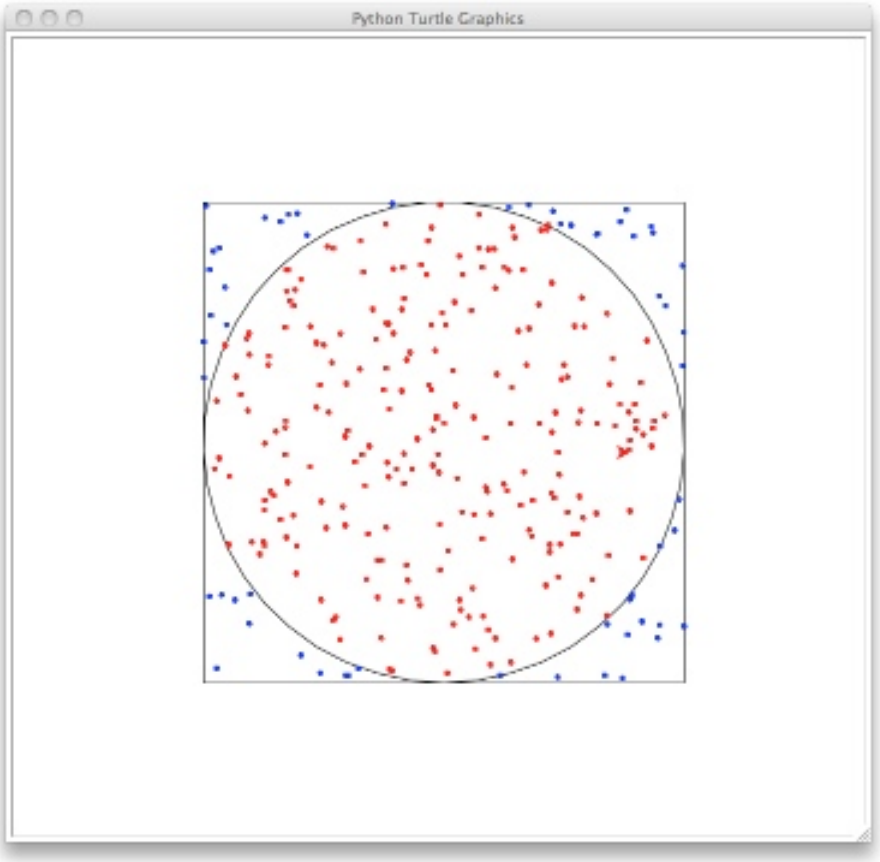
In class (on 10/3) we looked at how you can come up with an approximation for the value of pi using monte carlo simulation (generating random points). By generating random points in a 2 by 2 square and then counting how many fell inside a radius one circle inside this square, we could come up with an approximate for Pi.

Write a program that visualizes this approach using `turtle` graphics. The program should draw the outer bounding box, the inner circle and then generate random points within the square. If the point lies within the circle, it should be one color and outside the circle, another color. I strongly encourage you to start from the code we looked at in class and then add on from there.

You must implement the following function:

- `visualize_pi`: Takes as input two parameters, the radius of the circle and the number of samples to be drawn. Besides defining the size of the circle, the radius should also be used to define the dimensions of the outer bounding box and the range of the random points.

For example, below is a sample output from a call of `visualize_pi` with a radius of 200 and 300 points:



I encourage you to include other functions if appropriate. In addition to your code for this program, also submit a printed picture of an example run. See the submission requirements for Lab 2 for instructions on how to get a screen capture.

Possible program extensions: add some text to the turtle printout (look at the documentation for how to do this), for example, print out the approximation of pi or add a title, make the picture better looking.

When you're done

You should have three separate files/programs, one for each of the programs above. Make sure that each program is properly commented:

- You should have comments at the very beginning of each file stating your name, course (including section number), problem number and the date.
- Each function should have an appropriate *docstring*
- Other miscellaneous comments to make things clear

Print out all three of your programs as well as your picture for visualizing pi.

Grading

		points
style/comments (per program)		8 * 3
	if/while/for	
	variable naming	
	comments/docstrings	
	formatting	
	parameters	
	additional functions	
Taxes		
	`salary_calculator`	3
	`tax_bracket`	3
	`calculate_taxes`	2
	User input and program output	4
Word play		
	`ispalindrome`	4
	`vowel_count`	3
	User input and program output	5
Visualizing pi		
	outer square	2
	inner circle	2
	random points	4
	colored appropriately	4
Required extra add-ons		2
extra credit		3
total		62 (+3)