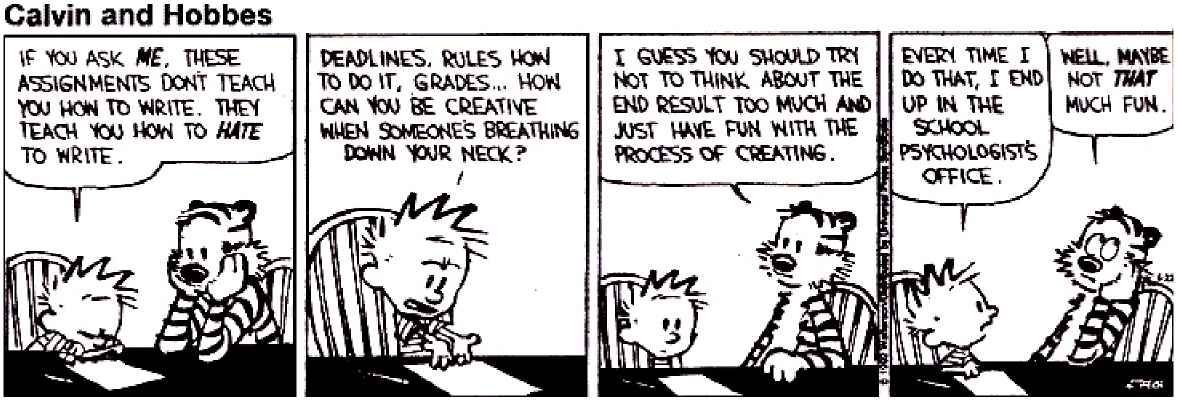
CS150 - Test Project 1

Try to have it done before break, but ...

Due: Wednesday Oct. 24, before the beginning of class


Calvin and Hobbes
IF YOU ASK ME, THESE ASSIGNMENTS DONT TEACH YOU HOW TO WRITE. THEY TEACH YOU HOW TO HATE TO WRITE.
DEADLINES, RULES HOW TO DO IT, GRADES... HOW CAN YOU BE CREATIVE WHEN SOMEONE'S BREATHING DOWN YOUR NECK?
I GUESS YOU SHOULD TRY NOT TO THINK ABOUT THE END RESULT TOO MUCH AND JUST HAVE FUN WITH THE PROCESS OF CREATING.
EVERY TIME I DO THAT, I END UP IN THE SCHOOL PSYCHOLOGIST'S OFFICE.
WELL, MAYBE NOT THAT MUCH FUN.

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 use the textbook, 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 not allowed**. 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 or submitting 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 stated below, the maximum number of points you can get is 60. In order to get the full credit of 63, 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 66). 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 MiddKid Support

As you talk to some of your friends around campus, you realize there is a real need for a program that calculates a student's GPA both for their current grades as well as for their overall time here at Middlebury. When you run the program:

1. The user is first prompted to enter their current GPA and also how many semesters they've been here (not including the current one).
2. The user is then prompted to enter grades for their four courses for the current semester.
3. The program outputs the GPA for this semester
4. and a revised version of their cumulative GPA that includes this semester.

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

```
*****  MiddKid App  *****
What is your current GPA? 3.0
How many semesters have you been at Middlebury? 4
Enter grade for course 1: A
Enter grade for course 2: B
Enter grade for course 3: b
Enter grade for course 4: A

------  Results  ------
Your GPA for this semester is: 3.5
Your new cumulative GPA is: 3.1
```

You **must** write the following functions:

- `letter_to_gpa` which takes a letter grade as a parameter and returns the GPA equivalent of that letter grade. The user function must handle both uppercase and lowercase letters, but it does not need to handle "+" or "-" variations of the grades.
- `semester_gpa` which takes no parameters. This function should prompt the user to enter their grades for their current courses (like above) and then should return the GPA for this semester. To get full credit you *must* use a `for` loop.
- `cumulative_gpa` which takes three parameters: 1) the current GPA, 2) the number of semesters the student has been at Middlebury (not including the current semester), and 3) the GPA of the current semester. The function should return the new cumulative GPA by combining the GPA from the previous semesters with the current semester. Make sure to do a weighted average.

Possible program extensions: Include support for “+” or “-” grade variations (you may **not** simply add in more `elif` statements for every combination), prompt the user for an optional 5th course, prompt the user for the number of courses, have the user enter a target GPA and calculate what grade average they must achieve on average for the remaining semesters to achieve this.

2 Vowels for Everyone

I LIKE TO VERB WORDS.
WHAT?
I TAKE NOUNS AND ADJECTIVES AND USE THEM AS VERBS. REMEMBER WHEN "ACCESS" WAS A THING? NOW IT'S SOMETHING YOU DO. IT GOT VERBED.
VERBING WEIRDS LANGUAGE.
MAYBE WE CAN EVENTUALLY MAKE LANGUAGE A COMPLETE IMPEDIMENT TO UNDERSTANDING.

Write a program that prompts the user to enter a word and then displays the following facts about the word:

- the number of vowels in the word
- whether or not the word has more vowels than consonants
- whether the word starts and ends with the same character
- the vowels in the word

For example:

```
Enter a word: banana

Banana has 3 vowels in it,
has more consonants than vowels,
and does not start and end with the same letter.
The vowels in banana are: a
```

or

```
Enter a word: area

Area has 3 vowels in it,
has more vowels than consonants,
and starts and ends with the same letter.
The vowels in area are: ae
```

To accomplish this you *must*

- Follow the output language and format above exactly.
- Capitalize the word at the beginning.
- Write a function called `count_vowels` that takes a word and returns the number of vowels in the word.
- Write a function called `more_vowels_than_consonants` that takes a word and returns whether or not that word has more vowels than consonants.
- Write at least one other function.

Possible program extensions: print out another fact about the word, print out all the unique consonants in the word, allow the use to repeatedly enter words until some stopping criteria is reached.

3 Visualizing pi

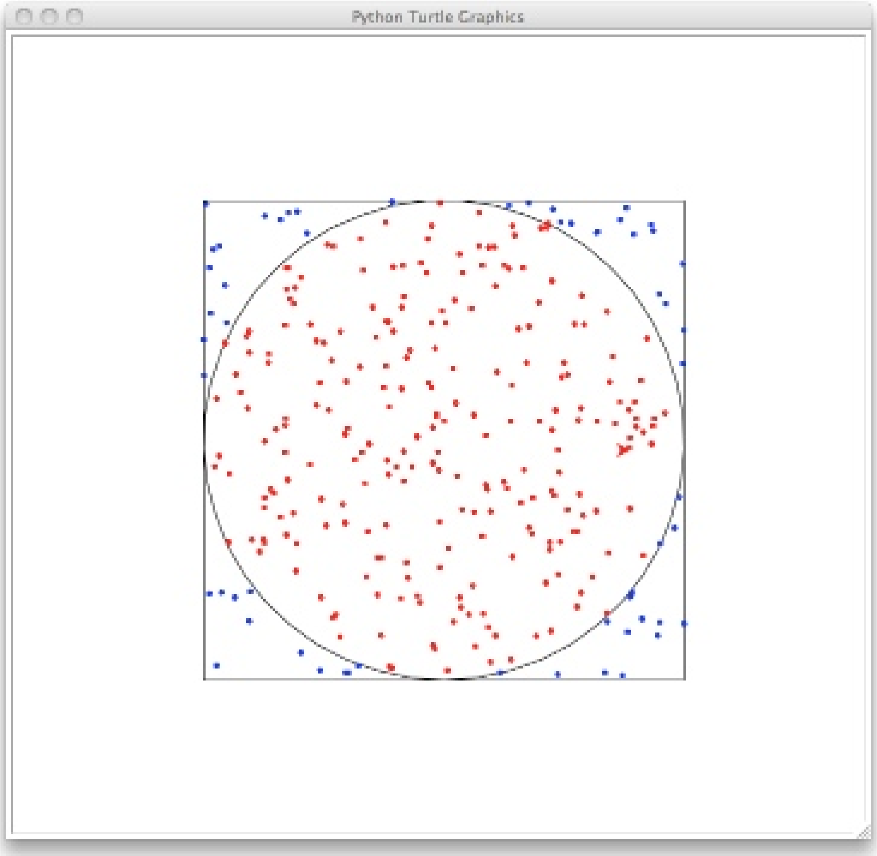
In class (on 9/30) we looked at how you can come up with an approximation for the value of pi using monte carlo simulation (generating random points). By generating random points in a 2 by 2 square and then counting how many fell inside a radius one circle inside this square, we could come up with an 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 a function called `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.
- You must implement at least one other function.

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 look better.

When you're done

You should have three separate files/programs, one for each of the programs above. The first should be called `middkidd.py`, the second `vowels.py` and the third `pi_visualize.py`.

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), test project number and the date.
- Each function should have an appropriate *docstring*
- Other miscellaneous comments to make things clear

In addition, make sure your program uses good programming style such as proper naming conventions, good use of functions, proper use of bools, etc.

Submission

Submit your file digitally online. To do this, you'll need to create a single file as follows:

- Create a folder with your name on it, e.g. I would create a folder and call it "kauchak".
- Put all three of your .py files, one for each of the problems, in the folder.
- "zip" up the folder (i.e. create a .zip file from the folder)
  - **Mac:** Right-click on the folder (i.e. ctrl + click) and select "compress".
  - **Windows:** Right-click on the folder and select "Send to" then "Compressed (zipped) folder"
- Upload this final .zip file digitally as usual. Enter "TP1" for the assignment number field.

Grading

		points
style/comments (per program)		8 * 3
	if/while/for	
	variable naming	
	comments/docstrings	
	formatting	
	parameters	
	additional functions	
	misc	
MiddKid		
	`letter_to_gpa`	3
	`semester_gpa`	3
	`cumulative_gpa`	2
	User input and program output	4
Vowels		
	`count_vowels`	3
	`more_vowels_than_consonants`	2
	start/end check	1
	vowels in word	3
	User input and program output	3
Visualizing pi		
	outer square	2
	inner circle	2
	random points	4
	colored appropriately	4
Required extra add-ons		3
extra credit		3
total		63 (+3)