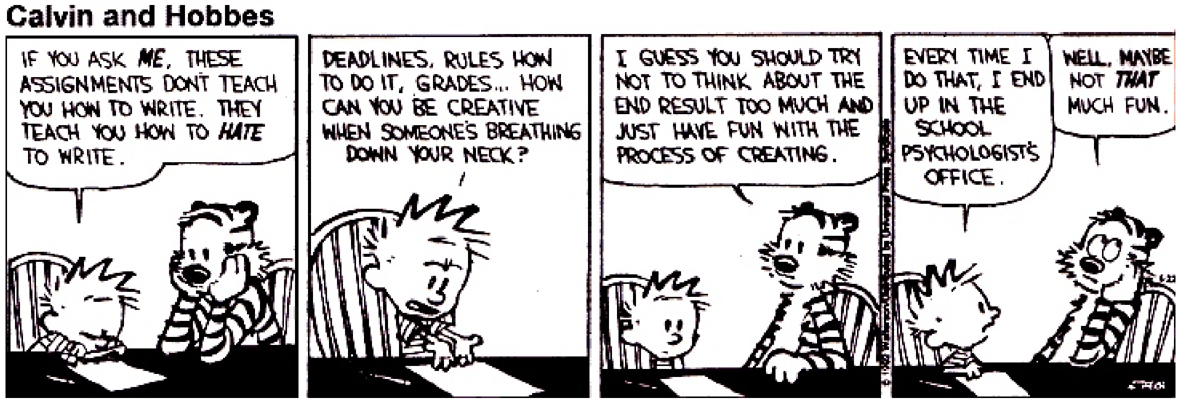
CS150 - Test Project 1

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

Due: Monday March 31, by 11:59pm


Calvin and Hobbes
IF YOU ASK *ME*, THESE ASSIGNMENTS DON'T 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 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.

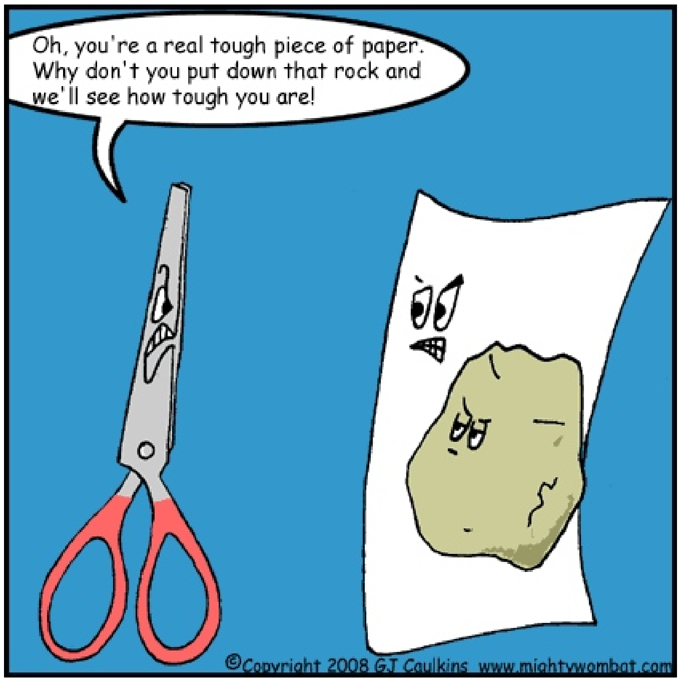
Complete each of the three problems below, each as a separate program, in a separate file. You are encouraged to reuse code from your assignments or our class examples. Partial credit will be awarded, so if you can't solve a whole problem, get as far as you can.

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 2 points of additional extra credit (i.e. for a total of 65). You may receive at most 3 additional points 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 Rock-Paper-Scissors


Oh, you're a real tough piece of paper. Why don't you put down that rock and we'll see how tough you are!
©Copyright 2008 GJ Caulkins www.mightywombat.com

We're going to implement a basic version of rock-paper-scissors. If you're not familiar with the game, see the Wikipedia page:

`http://en.wikipedia.org/wiki/Rock-paper-scissors`

The end result is a function called `play_game` that prompts the user for a selection of "rock", "paper" or "scissors", picks a random choice for the computer and then prints out the result of the game. For example, here are a few example runs of the function:

```
>>> play_game()
Enter rock, paper or scissors: rock
Computer picked: rock
Tie!
>>> play_game()
Enter rock, paper or scissors: scissors
Computer picked: paper
You win!
>>> play_game()
Enter rock, paper or scissors: paper
Computer picked: scissors
You lose!
```

To accomplish this you **must** write the following functions:

- `get_random`: Doesn't take any parameters but returns a random string of either "rock", "paper", or "scissors".
- `user_wins`: Takes two strings as parameters, the user's choice and the computer's choice, and returns `True` if the user wins and `False` otherwise. Notice that tying then would be `False` because the user doesn't win. For example (assuming the user parameter is the first):

```
>>> user_wins("rock", "scissors")
True
>>> user_wins("rock", "paper")
False
>>> user_wins("rock", "rock")
False
```

- `rock_paper_scissors`: Takes a string as a parameter, the user's choice, and plays a single game of rock-paper-scissors by picking a choice for the computer then printing out the results of the game. This function must use the previous two functions. For example:

```
>>> rock_paper_scissors("rock")
Computer picked: paper
You lose!
```

- `play_game`: Doesn't take any parameters but prompts the user to enter a choice then uses this choice to play the game. You may assume that the user enters a valid choice (i.e. "rock", "paper" or "scissors").

Ideas for possible program extensions: check to make sure that the user enters a valid choice (if they don't, don't play the game, or better, ask them to enter another choice), make the program loop until the user says they don't want to play the game anymore, use turtle graphics to draw the user's and computer's choices.

2 Vocabulary Game

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 an interactive program that when you “run” the program it starts playing the word game:

```
Would you like to play a word game? yes
How many seconds do you want to play for? 20
Enter a word that has two 'm's: mammal
Great!
Enter a word that has two 'm's: camel
camel doesn't have two 'm's in it
Enter a word that has two 'm's: mountain
mountain doesn't have two 'm's in it
Enter a word that has two 'm's: marmaduke
Great!
You named 2 words that have two 'm's
```

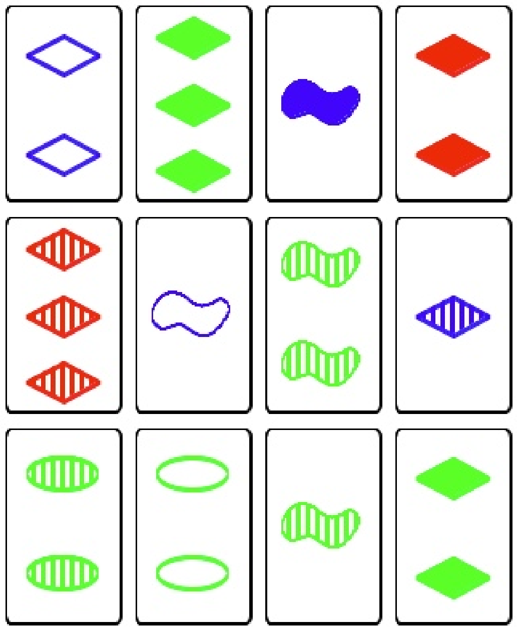
- Your program should follow the format above precisely.
- You must use at least 2 functions in your program.
- When you ask the user if they want to play the game, if they say any capitalization variant of “yes” (e.g. “YES”, “Yes”, “yEs”, ...) then the game should continue. If they enter anything else, then the game should just end. (Hint: there’s a straightforward way of doing this without having to check all the possibilities.)
- The letter should be randomly selected each time the game is played.
- Your program must check if the letter occurs twice in the word, though it does NOT need to check if it’s an actual word. If the letter occurs twice (or more) in the entered word, then a positive message should be printed. If not, the user should be told that the word does not have the letter in it twice.

I’m giving you a fair amount of flexibility about how you break this program into different functions. Think about the best way to break this into smaller components and use good style!

If you get stuck on checking if the word has two copies of the letter in it you may submit a version of the program that only checks if the letter occurs in the word once, but you will lost 1.5 points.

Ideas for possible program extensions: check if the word is in the dictionary (see the class notes on 3/12 for a list of English words); rather than one point per correct word, assign points based on the length of the word; offer the player an alternative version of the game (in addition to the original) that times how long it takes to list some predetermined number of words; add other constraints to the words entered besides just the first letter constraint

3 Set Game



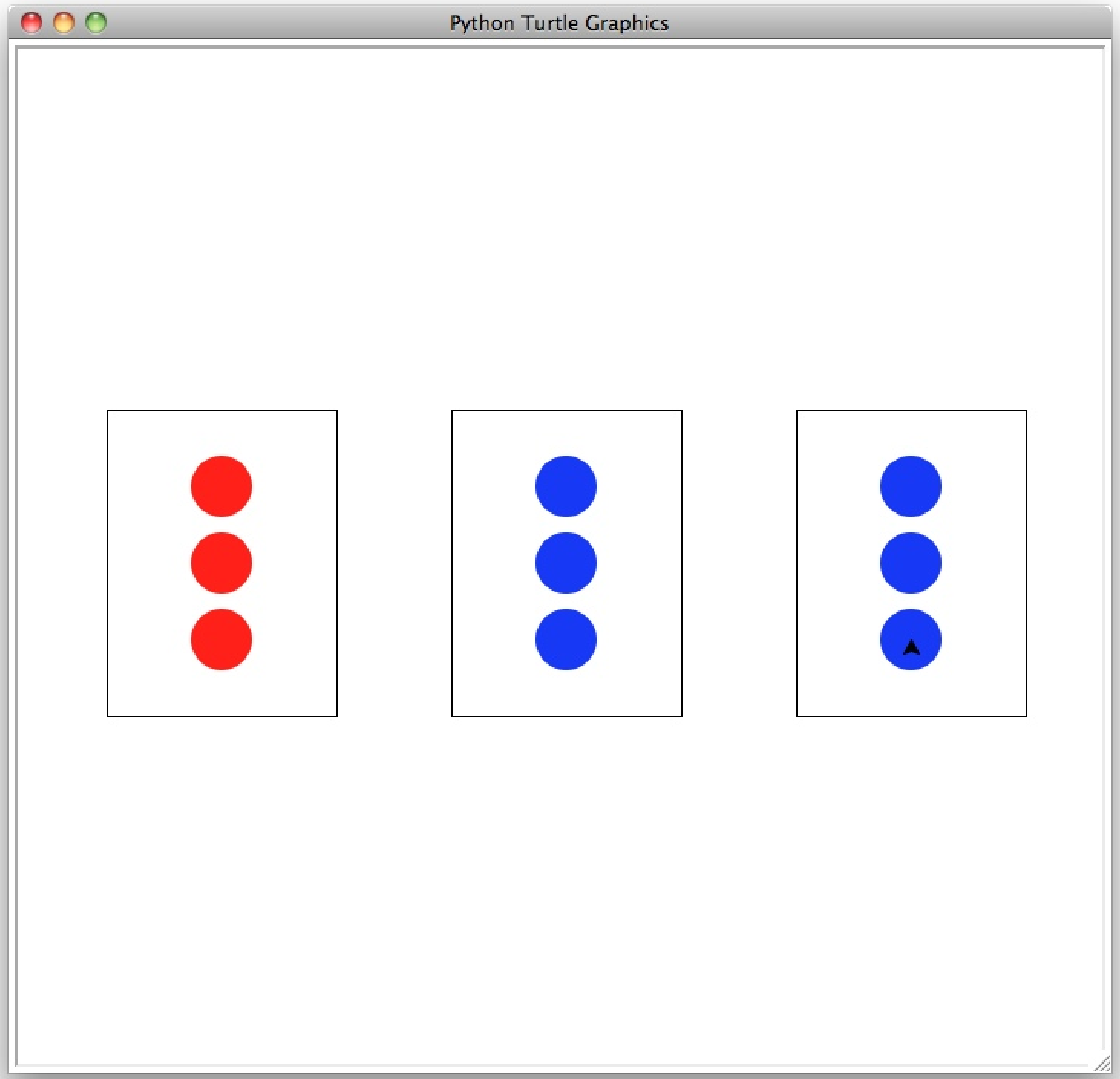
Set is a card game that starts with 12 cards laid out on the table. The set cards have a number of attributes including color, type of shape, number of shapes and the texture of the shape. A "set" is a collection of three cards where each attribute is either the same (e.g. all red) or all different (1, 2, 3 shapes). See

`http://www.setgame.com/set/rules_set.htm`

for the actual rules, examples, etc.

We're not going to implement this full game, but are going to get the ball rolling by writing a program that will generate some Set-like cards randomly using `turtle` graphics. Specifically, you should write a program that generates three cards evenly spaced on the screen. The cards will all have three circles/dots for the shape and each card will have a randomly chosen color picked from: blue, red and green.

For example, here is a sample output from a run of the program:



- You must implement a function called `generate_random_card` which takes two parameters, the x and y location of the top left corner of the card, and draws a card with three circles/dots with a random color.
- You must implement at least one other function to help you draw the three cards.
- You must use at least one `for` loop somewhere in your code.

It may be easier to use the `dot` function, instead of the `circle` function, though you may use either. See the `turtle` documentation for using `dot`.

Possible program extensions: generate 9 (or 12) cards instead of three (if you do this, you must use one or more loops to generate the cards); add more card variations, such as shape, texture and number of circles; add a caption to the picture (see the `write` function)

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

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	
Rock-Paper-Scissors		
	`get_random`	2
	`user_wins`	4
	`rock_paper_scissors`	4
	`play_game`	2
Vocabulary Game		
	play game or not	2
	timing works	2
	random letter	1
	check entered words	2
	proper counting	1
	proper output	2
	used 2 functions	1
	general functionality	1
Set Game		
	card placement	3
	circle placement	3
	random color	2
	use `for` loop	1
	use another function	1
	`generate_random_card`	2
Required extra add-ons		3
extra credit		2
total		63 (+2)