Goals

- Practice with some old and some new Python concepts and tools.
- More practice with Python lists
- More practice with Python coding and testing.
- Combining concepts and tools in a variety of ways.

Complete problems (0), (1), and (2), plus ONE problem from (3) through (7).
Challenge: Complete more than one problem from (3) through (7).

(0) Testing and Debugging/Lists/Double Preceding

Your lab partner claims to have written a function that replaces each value in a non-empty list with twice the preceding value, and the first value with zero.

```python
def double_preceding(values):
    '''(list of ints) -> None

    Update each value in a list with twice
    the preceding value, and the first item
    with 0.

    For example, if x has the value
    [1, 2, 3, 4, 5], after calling
    double_preceding with argument x,
    x would have the value [0, 2, 4, 6, 8]
    '''

    if not (len(values) == 0):
        temp = values[0]
        values[0] = 0
        for i in range(1, len(values)):
            values[i] = temp * 2
            temp = values[i]

    return #None

(a) Design several test cases to test double_preceding, including simple examples, examples that test all of the paths through the function, and boundary examples.

(b) As a result of (a), you will notice that double_preceding does not work as advertised. Fix the code and re-run the test examples from (a).
(1) Password Protection
SecuriCorp has recently been the victim of a number of security breaches. Internal analysis has determined that employees use simple passwords that are too easy to guess. You have been hired to write a password checking program. This program should contain a function passwordChecker which takes a password and returns True if it is sufficiently secure. Otherwise, it should return False. To ensure security, management has come up with the following security regulations:

- Passwords must be at least 5 characters long
- Passwords must contain at least one upper case letter
- Passwords must contain at least two numbers
- Passwords may not contain the characters "E" or "e"
- Passwords must not contain the characters "E" or "e"
- A Password may not be a frequently used password: password, Two34!, qwerty, letmein, trustno1, 111111, passw0rd

Consultants suggest writing a separate function to test a password against each of these conditions. (Hint: reuse – with some changes - code from previous projects.) These functions could then be called from the main passwordChecker function. However, implementation details are left to you.

Write the code (passwordChecker function and auxiliary functions) and execute it for a sufficient number of test cases that SecuriCorp will be confident of their passwords, including at least the following:

```python
>>> passwordChecker('111111')
False
>>> passwordChecker('#Qwerty')
False
>>> passwordChecker('#qwErty')
False
>>> passwordChecker('#Qw9rty')
False
>>> passwordChecker('#Qw9rty')
True
>>> passwordChecker('Two34!')
True
>>> passwordChecker('A99!')
False
>>> passwordChecker('Abyz9!')
False
>>> passwordChecker('qwerty99!')
False
>>> passwordChecker('Qwerty99')
False
>>> passwordChecker('')
False
>>> passwordChecker('OK99!!')
True
>>> passwordChecker('$100abC')
True
```
Challenge: For passwords that are not secure, print out a message telling why (one message per password).

(2) Files: Using a simple text editor (for example, the IDLE editor), create a file (of favorite foods? movies? important dates?) and save it. Then write a Python function, processfile, which opens the file, creates a list of each item in the file, sorts the list, and prints it. None value is returned.

For example, for a text file that lists the days of the week:
Monday
Tuesday
Wednesday
Thursday
Friday Saturday Sunday

>>> processfile('project9.txt')
Friday Saturday Sunday
Monday
Thursday
Tuesday
Wednesday

Choose ONE or more of problems numbered (3) through (7):

(3) Exploring Python - based on GCM, 6.6.2: Calendars

(a) In the Python Shell, import Python module calendar. Use the dir function

>>> dir(calendar)

or autofill

>>> calendar.

to see a list of what the calendar module contains.

Using function help, investigate the calendar module functions.

(b) Using calendar module, write a function with three parameters, date (1-31), month (1-12), and year, and returns the day of the week (a string) that the input date falls on. (Assume correct and well-formed input for the date, month, and year arguments.)

>>> whatDay(9, 6, 2014)
'Monday'
(4) **Revisit Mars Explore**

(a) Edit `mars_explore_main` function to have one parameter, n, which is the number of times to call `mars_explore`.

(b) Edit `mars_explore` to return a list of two values, water content and temperature.

(c) Edit `mars_explore_main` to have a list of all of the water content values and another list of all of the temperature values returned by the `mars_explore` expeditions.

(d) Edit `mars_explore_main` to determine and report the average water content and the average temperature after all of the calls to `mars_explore`.

(5) **Files/urlib**: Use Python `urllib` to access one or more web page(s). Then print each line of the web page (HTML code) and/or search the web page and report and/or process data from the web page. (If you don't have a particular web page in mind, try data.gov, data.cityofeugene, electoral-vote.com, or websites listed at [http://blog.visual.ly/data-sources/](http://blog.visual.ly/data-sources/).) None value is returned.

(6) **Grade calculator**

(a) Update the `grade_calculator` function from project 5 so that it has two parameters, `projects` and `exams`, each of which is a list of lists of scores and adjustment factors. Note that the parameter lists mean that the user will no longer be asked to enter scores. For example:

```python
grade_calculator(
    [[25, 1.17],[38, .875],[50, .583],[35, .875],[60, .5]],
    [[39, .4],[40, .5]])
81 B
```

(b) Then update the `grade_calculator` function so that the `projects` list is a list of 9 scores, of which the top 7 are included in the final grade calculation. Each of these 7 is worth a maximum of 3.5 points, and an additional .5 is added to the total projects score. Last, exams should be a list of 3 exam scores. The last exam will be graded out of 60 points and is worth 30 points in the final grade calculation.

(7) **Art Show**

Start by writing a function, `art_show`, to have the turtle draw a simple line drawing of anything you like (for example, a house – see project 4). Then add details. Parameters are up to you; `art_show` returns the value None. Test the function:  ```python
>>> art_show()
```
Grading Rubric

This project will be marked out of 70 points. Problem (0) is worth 10 points: 6 points for good tests (function calls in the docstring) and 4 points for fixing the bug/correct code. Problem (1) is worth 20 points: 4 points for the docstring type contract and function description; 16 points for the main function code including the return statement. Problem (2) is 15 pts: 6 points for the docstring (type contract, function description, example), 5 points for code including the return statement, and 4 points for correct output. The choice problem is worth 25 points: function header (4 pts.), docstring (6 pts.), well-written code (7 pts.), return statement (2 pts.), and correct output (i.e., working code) (6 pts).

Getting Started

Write your program using an editor window, so you can save your file.

You do not need to turn in the results of running your code.

Programming style is important! Remember
- Include a docstring in every function
- Use whitespace between operators and operands
- Use descriptive variable names
- Add appropriate comments.

Finishing & submitting your work

When you have completed all of the problems, add additional comments to your code to make sure the functions for each problem are clear. Check the list of functions above to make sure that your Python file includes all of them. Do a final Save command to save the code in the editor window as a file with the name cis122project9.py.

To submit your project, login to Blackboard. From the menu on the left hand side of the screen, choose "Projects". In the projects folder, choose "Submit Grab Bag". In Section 2 of the page that is displayed, scroll down to "Attach file" and choose "Browse My Computer". Locate cis122project9.py (the file you just created), and double click on it. The file name will appear on the "Attached Files" list.

At the bottom of Section 2, you will see a "Comments" window. This is where you credit all of the sources of any help you may have received on this assignment, including your partner if you are working in a programming pair. This is also the place to include any feedback you may have about the assignment and/or any remaining questions you may have.

Scroll down to Section 3 and hit the "Submit" button. You may re-submit your project up until the project deadline. Only the final submission will be graded.