Goals

By the end of this assignment, you will have

- more practice with exploring Python modules
- more practice with a tricky concept: return v. print
- more practice with functions calling functions

As always, every function should reflect the function design recipe function, by including a docstring with a type contract, function description, and examples of function calls. Every function should include a return statement.

Remember to include from turtle import * at the top of your Project 4 Python file (in the editor window).

(0) Sunshine on a Cloudy Day

The diameter of the Sun is 1,392,000 km, while the equatorial diameter of the Earth is 12,756 km (a ratio of about 109:1). Using Turtle Graphics, write a function, sun_and_earth, to illustrate the relative size of the sun and the earth by drawing two circles. sun_and_earth has no parameters and should return the value None.

Look at turtle builtin function circle. Some other turtle functions that will be helpful are begin_fill(), end_fill(), and fillcolor(). When begin_fill() is called, the turtle keeps track of its starting point, and all the lines it has drawn until end_fill() is called. When end_fill() is called, the turtle fills in the space enclosed by the lines that the turtle has drawn.

Other turtle functions that may be helpful here are setposition() and up() and down(). penup() and pendown() lift the turtle “pen” from the canvas and put it back down, respectively. setpos will move the turtle directly to the x,y position specified by its arguments. (That is, the setpos args specify an absolute position, not relative movement. Explore the functions using IDLE’s help function or using the Python documentation.

(1) More Turtle Graphics

(1a) Write a function, square, to draw a square, using turtle movement commands (e.g., fd, bk, lt, rt). Function square should have one parameter, length, which is the length of the sides of the square. square will return the value None. Test function square:

```python
>>> square(100)
>>> square(200)
```

(1b) Edit function square by adding a second parameter, scolor. The function should now draw an scolor-colored square.

Test function square with whatever arguments you like, for example:
(1c) Write a function, triangle, with two parameters, length and tcolor, which will draw a tcolor-colored triangle when called. Triangle will return the value None.

```python
>>> triangle(100, "orange")
>>> triangle(150, "purple")
```

(As in square, the turtle will need to rotate 360 degrees to complete the triangle and return to its initial heading. In square, the turtle made four turns; for a triangle it will be three.)

(1d) Write a function, house, that draws a square house with a door and a roof. The house body, door, and roof should be drawn by calling functions you have already defined. The house function definition will not have any parameters and will return the value None.

Test the function:  >>> house()

(2) Exploring Mars: The Python “turtle” is essentially a robot that is controlled with Python code. Write a program that will send a robot to conduct a short exploration of the surface of Mars. You will use a method from the random module, which means you will need to add code to import random at the top of your Python file.

Start by entering the following function into your Python file. (Cutting and pasting from a .pdf is not recommended, but you can download this code from the class website or, probably best, type it in yourself.)

```python
def mars_explore_main():
    '''() -> None
    main function for mars_explore:
    set up print and graphical output
    then call mars_explore repeatedly
    
    data is printed; None value is returned
    
    >>> mars_explore_main()
    ''
    
    # label for print output
    print("xpos", "\t",
         "ypos", "\t",
         "water", "\t",
         "temp")

    # set up graphical output
    reset()
    title("Mars Rover")
    display_color = "blue"
    color(display_color)
    square(10, display_color)  # draw the rover
```
# explore five places on Mars
#mars_explore()
#mars_explore()
#mars_explore()
#mars_explore()
#mars_explore()

return #None

We will talk about this code in lab this week. You should also test it to make sure it works and read through it so you understand what is happening on each line.

Here are the specifications for writing the rest of the Mars Explore program:

(2a) Start by writing three functions: `rover_loc`, `water_content`, and `temp`. There are no parameters. Each of these functions will use the `randint` function from module `random` to simulate data collection on Mars. The `rover_loc` function should return a random integer between -275 and 275. The `water_content` function should return a random integer between 1 and 290 (ppm). And the `temp` function should return a random integer between -178 and 1 (degrees Fahrenheit).

For example:

```python
def rover_loc():
    '''() -> int

    return random number for rover location

>>> rover_loc()
125 [for example]
'''
    return random.randint(-275, 275)
```

Test each of these functions individually.

(2b) Write a function called `mars_explore` to do the following:

- Determine the next location for exploration. (Call `rover_loc` twice, one time to get the x coordinate and another time to get the y coordinate of the rover's next location.)

- Move the rover (turtle) to the new location and mark the position. (You may want to explore the `stamp()` and `dot()` functions, for example, or use your square or triangle.) Show the rover's path (i.e., move the turtle with the pen in the down position).

- Collect data at this location: assign local variables (i.e., variables defined inside the `mars_explore` function) to the values returned from calls to the `water_content` and `temp` functions.

- Display the data: use the Python print function to display the x and y coordinates of the rover, and the data for the water content and temperature at this location.
Function mars_explor has no parameters and returns the None value. Test your function after each step is completed. Do not wait until the end! When the function is completed, call it a few times to see different explorations.

(Challenge)

Display the temperature and water content data on the turtle canvas, as well as in the Python Shell. (Explore the turtle write function.)

(Challenge)

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). Then add details. Parameters are up to you; art_show returns the value None. Test the function:  >>> art_show()
Grading Rubric

This project will be worth 40 points (marked out of 80 points).

Functions sun_and_earth, square (with color), triangle, house, are each worth 10 points, with 2 points for the function header, 3 points for a docstring demonstrating use of the function design recipe (type contract, description, examples), 3 points for the body of the function including the return statement, and 2 points for correct output when the function is executed.

Functions water_content and temp are each worth 7 points, with 2 points for the function header, 3 points for the docstring, 1 point for the body of the function including the return statement, and 1 point for correct output when the function is executed.

Function mars_explore is worth 26 points: 2 points for the function header, 4 points for the docstring, 12 points for the body of the function including the return statement, and 10 points for correct output when the function is executed.

Getting Started

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

Programming style is important! Remember

- Include a docstring in every function per example in text p. 47
- Use whitespace between operators and operands
- Use descriptive variable names
- Add appropriate comments.

Examples are given for function calls. The examples help explain the function and also can be used to test the function. You do not need to turn in the example function calls.

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 and also rover_loc and mars_explore_main. Do a final Save command to save the code in the editor window as a file with the name cis122project4.

Call mars_explore_main() and take a screenshot of your Turtle Graphics output window and another one of the Python Shell where the output appears:

On a Mac – press cmd-shift-4, then press the space bar. The cursor will change to a camera. Move it over the turtle graphics output window (which will then be highlighted) and click the mouse button. The screen shot will be saved as a .png file on your desktop (or as a .pdf with Mac OS 10.3 and earlier). Or, press cmd-shift-4 to save the screen shot to the clipboard.

On a Windows computer – press alt-printscreen to save the active window to the clipboard. Open an application such as Word or Paint and paste the screenshot into a document; save the document.
Call `house()` (and `art_show()`, if you did the challenge) and take screenshots of the Turtle Graphics output window for those, too.

Zip the `cis122project4.py` and `.png` (or `.pdf`) files: find the files on your computer and select all (cmd-click or cntrl-click). When the files are selected, right click and choose Compress or Zip from the menu to create a `.zip` file. Rename the `.zip` file `cisproject4.zip`.

**IMPORTANT STEP:** If you resubmit your work, you must resubmit ALL of your files, zipped again. Then the grader needs to open only one submission per student. Thank you!

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 Hello, Turtles". In Section 2 of the page that is displayed, scroll down to "Attach file" and choose "Browse My Computer". Locate `cis122project4.zip` (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.
This is only an example. Every Mars Exploration will look slightly different. Your data may not appear on the graphical output (that was a challenge problem).

```
>>> mars_explore_main()
xpos  ypos  water  temp
-246  181   7     -136
255  -137  83   -96
-197  85   259  -100
-33  -173  106  -54
69  13   41   -46
```