# CS51A - Assignment 2
Turtle Graphics

Due: Sunday February 9, at 11:59pm

For this assignment, we will be using the `turtle` graphics module to draw a picture. You will have two options, a seascape with fish and rocks or a space scene with spaceships and planets (come talk to us if you have an idea of another scene type that uses similar shapes). Below is an example completed picture:
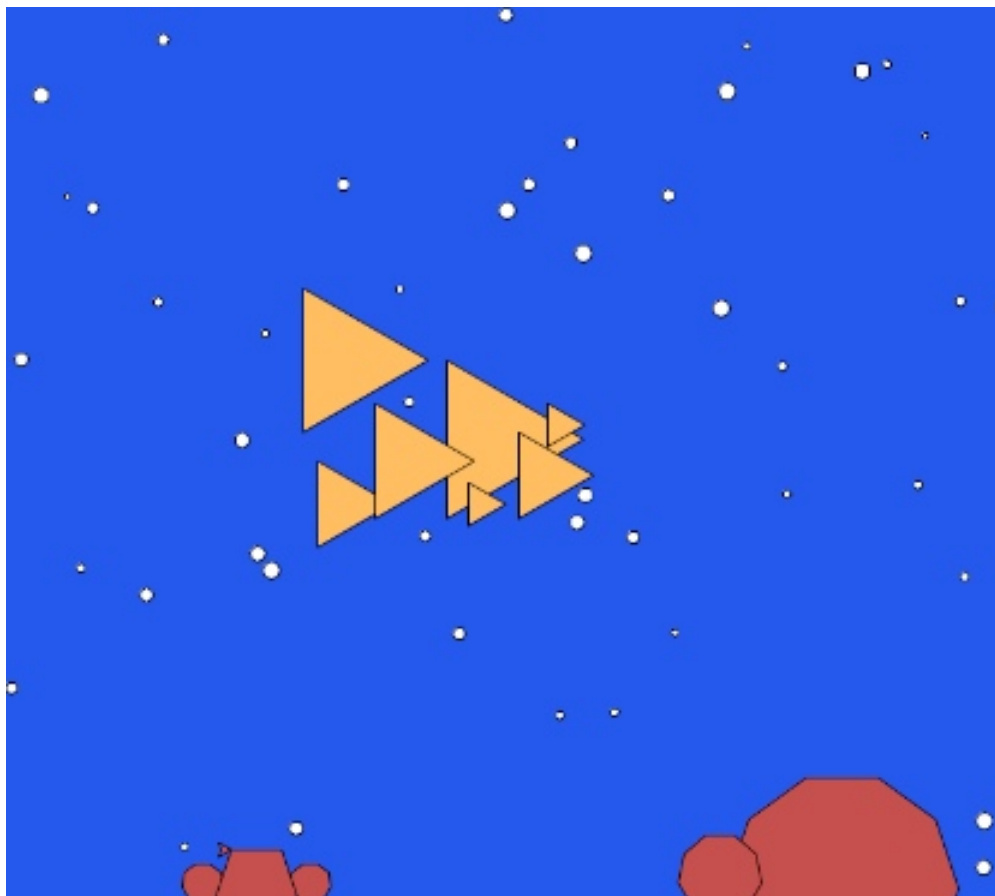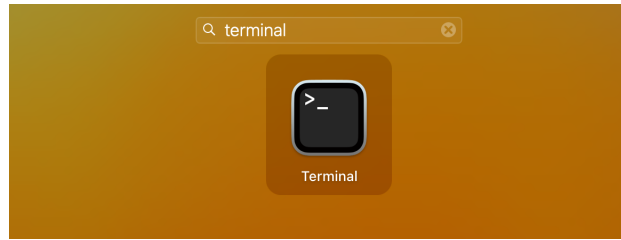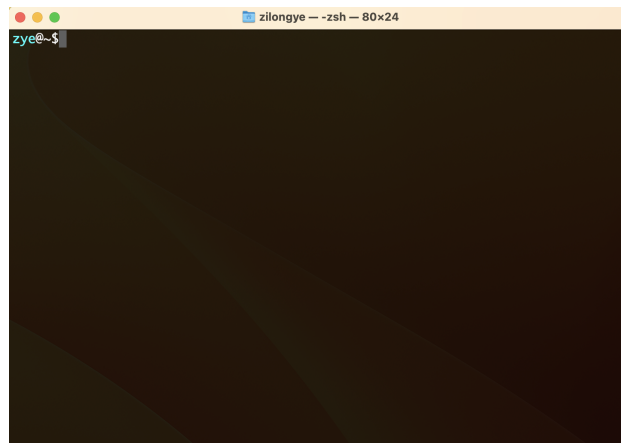
Figure 1: Search for 'Terminal' in Launchpad.

Figure 2: The 'Terminal' in Mac OS.

1 Command Line Interface

Before we start drawing the picture of our choice, we are going to take a detour and introduce the command line interface. Usually we interact with computers using a graphical user interface, for example, when creating a folder or copying/pasting a file, and other basic operations we introduced in Assignment 1. In addition to graphical user interfaces, there is a more traditional (and powerful) way to interact with computers and manage the computer resources: the command line interface. You can use a command line interface to perform the same operations (and many more!) you are used to in graphical user interfaces (e.g., move or copy/paste files).

1.1 Open up the command line interface

The command line interface is also known as the "Terminal". If you are using a Mac, you can open up the command line interface by clicking on your 'Launchpad' or 'spotlight search' and searching for 'Terminal' (see Figure 1) you will be able to see an application shown as below. Click on 'Terminal', then you will see the command line interface pop up as a new window (see Figure 2)

If you are using a Windows computer, you can open up its command line interface by typing in 'command' in the search bar and then clicking on the application named 'Command Prompt' (see Figure 3). You will be able to see a new window pop up (see Figure 4), which is the command line
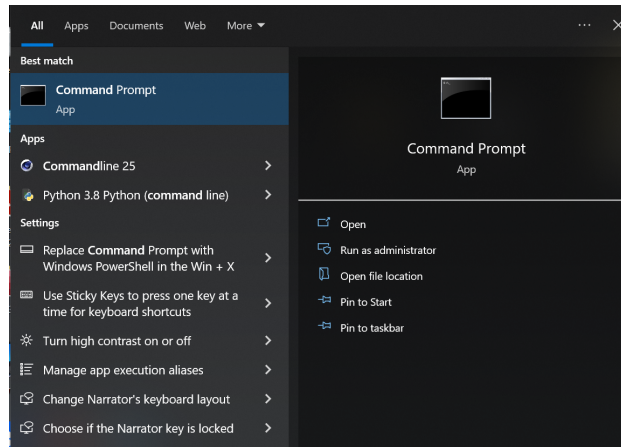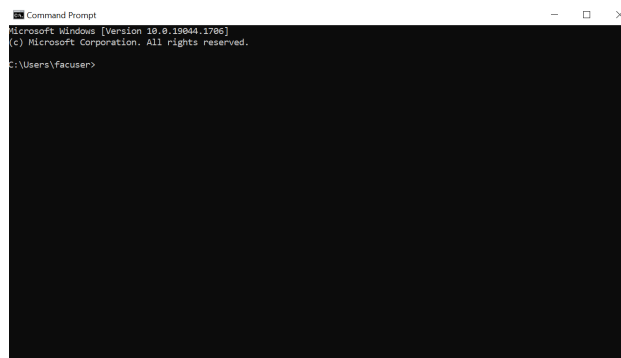
Figure 3: Search for 'Command Prompt'.

Figure 4: The 'Command Prompt' in Windows.

interface in Windows.

1.2 Navigating your computer

Just like using the graphical user interface, we can use the command line interface to navigate to any folder on your computer as well. Usually, when you open up the command line interface, you will be at the home directory by default (e.g., /Users/YOURACCOUNT on Mac, or C:\Users\YOURACCOUNT on Windows). You can check your working path or working directory (where you currently are) by typing in 'pwd' (Mac) or 'cd' (Windows) in your command line interface. After you type in the above command, your current working path will be printed (see Figure 5 for Mac OS and Figure 6 for Windows).

You can view the contents (files and directories) of the current working directory by typing in the command of 'ls' on Mac or 'dir' on Windows (shown in Figures 5 and 6).

If you want to navigate to a directory/folder in your current working directory, you can use the command 'cd' to achieve it. This is the same for both Mac and Windows. For example, if you
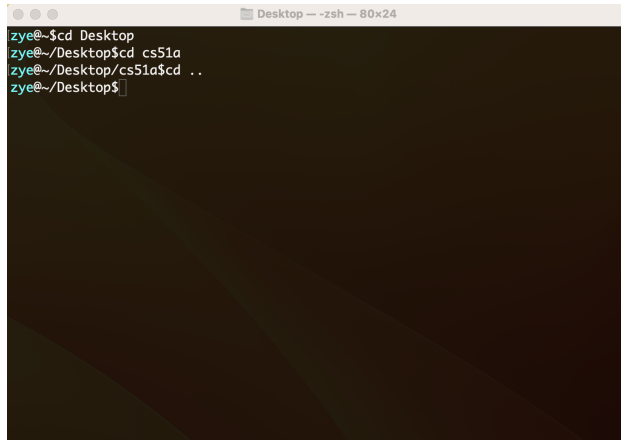
Figure 5: Print working path on Mac OS.

Figure 6: Print working path on Windows.

Figure 7: Navigating directories on Mac OS.
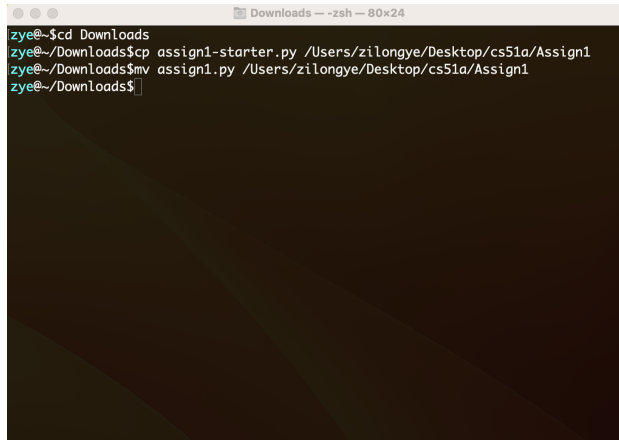
Figure 8: Navigating directories on Windows.

would like to go to the 'Desktop' folder, the command would be 'cd Desktop' (if 'Desktop' is in your current working directory). Note that there is a space between the command of 'cd' and the directory name you want to navigate to. If you would like to exit the current working directory and go back to the previous level of working directory, then the command to use is 'cd ..'. Note that there is also a space between the command of 'cd' and the two dots. Figures 7 and 8 show how to navigate on Mac and Windows.

When interacting with a command line interface, you will use the 'pwd' and 'ls' very often as you inquire what is your current working directory and what are its contents.
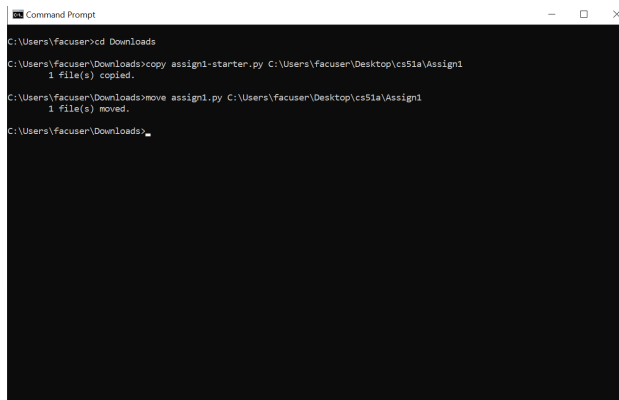
1.3 Move or copy/paste a file

You can use the command line interface to move or copy/paste a file from a source directory to a destination directory in a very convenient way. To do so, firstly, you need to know the working paths for both the source directory and the destination directory. Secondly, you need to use the 'cd' command (as we discussed in the previous subsection) to navigate to the source directory. To

Figure 9: Copying files on Mac OS.

Figure 10: Copying files on Windows.

copy a file (e.g., keep the file in the source directory and make a copy in the destination directory), you can use the command 'cp' on Mac or 'copy' on Windows. To move a file (e.g., move the file from the source directory to the destination directory without keeping it in the source folder), you can use the command of 'mv' on Mac or 'move' on Windows. The syntax to use the command line interface to move or copy a file is in the format of 'command filename destination. Note that there is a space between the command and the filename and the destination.

This assignment does not have a starter, but future assignments will (e.g., next asssignment). For example, to copy the file of 'assign3-starter.py' from 'Downloads' to my project directory 'Assign3' in the 'cs51a' folder on my 'Desktop', I used the 'cd' command to navigate to my 'Downloads' folder first. On Mac OS, I typed in the command 'cp assign3-starter.py /Users/XXX/Desktop/cs51a/Assign3' to perform the copy operation (here, XXX is your account name). On Windows, I typed in "copy assign3-starter.py C:\Users\XXX\Desktop\cs51a\Assign3" to perform the copy operation (here, XXX is your account name). From this example, you can see that it is important to know the working path of the source folder and destination folder. See Figures 9 and 10 for the aforementioned example.

Note 1: if you want to move or copy multiple files, you can use the same command format, but separating multiple files by spaces, e.g., 'command file1 file2 file3 destination'.

Note 2: to denote the hierarchical structure of the working path, Mac OS uses forward slash (e.g., /Users/XXX), while Windows uses back slash (e.g., C:\Users\XXX).

2   Picture Requirements

Now that we covered the basics of the command line interface, we are ready to dive into our assignment. We are going to give you a fair amount of flexibility in what you draw for this assignment. The key requirements are that your drawing must have:

- at least six fish/ships and six rocks/planets (though if you'd like to have more, that's fine too)

- a background (solid is fine, though other things would be great)

- bubbles/stars randomly placed on the background

Before actually coding, it's a good idea to plan out what you will be creating. On a piece of paper (ideally graph paper) plan out the design for your drawing. The screen width will be about 840 (ranging from -420 to 420) and the screen height approximately 787 (ranging from -393 to 393). With these dimensions in mind, plan out on paper where the different shapes will go, their sizes, etc. Note that the particular dimensions of the area do differ from computer to computer, so just work to get a rough outline and placement, but these may change a bit when you actually program it.

Things to think about:

- What is the x, y coordinate of the upper left hand corner of your fish/ships?

- What is the side length of each of the fish/ships?

- What is the x, y coordinate of the top of your rocks/planets?

- How many sides and what are the lengths of the sides of your rocks/planets?

3   Turtle documentation

A crucial programming paradigm is code reuse. Rather than writing your own code, say a `turtle` module, you can use one that someone has already written. An important part of coding then is documentation, both generating it for your programs and being able to understand other people's documentation.

The documentation for the `turtle` module can be found at:

`https://docs.python.org/3/library/turtle.html`

If you scroll down a bit, you can see an overview of all of the functions available to you.

To get ready for this assignment, you're going to need to read a bit more about some of the functionality of the `turtle` module. Below are a few methods that may be useful. Click on each of them in the documentation and make sure you understand what they do as well as what parameters they take and what, if anything, they return.

- `speed`
- `set_heading`
- `fill_color`
- `begin_fill`
- `end_fill`
- `penup`
- `pendown`
- `bgcolor`
- `goto`
- `bye`

Besides looking at the documentation, another way to get information about a function is with the `help` function. The `help` function takes as an argument the name of a function and outputs the docstring (i.e. function description). Start up `Python Console` then use `help` to get the documentation for `window_width` and `window_height`, which also may be useful for this assignment. Remember you will need to import the `turtle` module first by typing `from turtle import *` before trying to call help on any of the `turtle` module functions.

4   Style

Before you start coding, a few brief comments on style. We've talked about style components in class and now we're going to start utilizing these in your assignments. In particular, make sure you keep the following in mind as you're writing your program:

- All functions should have appropriate docstrings.
- Comment your code appropriately. Comment complicated parts of the code and include your name, date, etc. at the top of the file.
- Follow the variable naming conventions discussed in class and use good variable names.
- Use whitespace appropriately to make your program easier to read.

5 Basic Shapes

Create a new python file in PyCharm and give it the name `assign2.py`.

To get started, we're going to write some functions to make some basic shapes. Make sure that you have these working before moving on to the next part.

Don't forget to include the import statement for the `turtle` module at the top of your file:

```
from turtle import *
```

- **triangle** - Write a method called `triangle` that draws an equilateral triangle (i.e. a triangle with all three sides the same length). Your function should take three parameters: the x and y coordinate where it should be drawn and the length of the side of the triangle. The triangle should be drawn so that the left edge of the triangle is vertical. For those rusty on geometry, the interior angles of an equilateral triangle are all 60 degrees, which means the angle between a straight line drawn from one side and the adjacent side is 120 degrees.
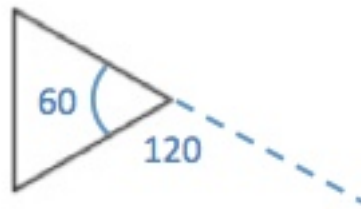
  Your x and y coordinates may indicate any part of the triangle (e.g. the top left, the center, etc.). Pick whichever seems easiest.

  For example, here are three triangles of increasing size:
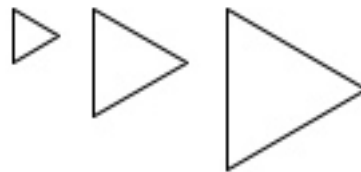
- **polygon** - Write a method called `polygon` that draws a polygon. An n-sided polygon has n equal length edges and the angle between a straight line drawn from one side and the adjacent side is $360/n$. Your function should take 4 parameters: the x and y location of the polygon, the number of sides, and the length of the each side. The top edge of your polygon should always be perfectly horizontal. Again, the x and y coordinates may indicate any point on the polygon, so pick whatever seems easiest.

Unlike the triangle where you know when you're writing the function exactly how many sides the object will have, for the polygon you can't just hard-code the different line segments. Instead, you'll have to use a `for` loop.

For example, below are some polygons of differing number of sides and size. All can be drawn with the `polygon` function.
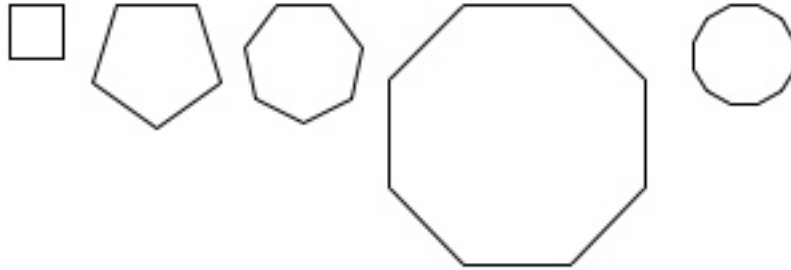
6   The Background

At this point, you should have two functions written that draw triangles and polygons anywhere on the screen. We'll now work on filling in the background.

The key component of the background is randomly spaced circles. Write a function called `add_circles` that takes as a parameter the number of circles to add and randomly places circles of radius 4 throughout the screen. Some hints for how to do this:

- The `circle` function draws a circle with a given radius.

- You'll need to figure out the dimensions of the screen (look at Section 3 again if this doesn't seem familiar).

- The `randint` function from the random module may be useful. To make this available, don't forget to include `from random import randint` at the top of your program.

To check that everything is working right, try adding differing numbers of circles and make sure that they're distributed throughout the screen and that the right number are actually being drawn.

Once you're sure it's working, add a bit more code to make the size of the circles random. You'll have to play with different size ranges to see what looks best. For example, if you look at the picture on the first page of this handout, you'll see that the bubbles range in size.

7   Putting it all together

You now have all the components required to put together your final picture. Create a function called `generate_picture` that draws the entire picture. One way to do this is:

1. Change the background to the appropriate color.

2. Change your `triangle` and `polygon` functions so that they create filled shapes (see the `turtle` documentation on how to do this if you don't remember).

3. Change your `add_circles` method to also draw filled circles.

4. Actually create your picture using your three functions. Your picture should include both triangles and polygons as well as some randomly filled circles using your `add_circles` function. I encourage you to get creative here!

5. Change the fill color of the objects to make it look more realistic (e.g. orange fish and brown rocks).

8   More practice

Write a function called `rectangle` that draws a colored rectangle. The function should have four parameters: the x and y location of the rectangle, the height, and the length. The rectangle should be drawn so that the top edge is flat and represents the "length" side of the rectangle. The rectangle should be filled in with one of three colors (you can choose the three colors), based on the following conditions:

color 1: if the rectangle is a square

color 2: if the rectangle is longer than it is tall

color 3: if the rectangle is taller than it is wide

Note that these three cases fully capture all possible cases for a rectangle, so it should be colored one of these three colors.

For this part, you just need to write the function. If you want to incorporate it into your picture, great, but it is not required.

9   Extra Credit

You can earn up to 1 point of extra credit on this assignment. Extra credit will be awarded by including additional elements to your picture. The amount of points given wiwel be assigned based on creativeness and difficulty of implementation. Here are some examples, but we encourage you to include your own:

- Instead of circles, write a function called `star` that draws stars (not asterisks) (must use a `for` loop)

- Include other types of objects in your scene

- Modify the `triangle` function to draw more interesting triangles, for example isosceles triangles or triangles with fins

To receive extra credit you MUST include in your comments at the top of the program what the extra credit additions you added were (otherwise, it can be hard to figure out sometimes).

10 Ethics

ChatGPT is changing many things, including colleges and universities. In Canvas under the "Files" tab there is a file call `chatgpt_artificial_intelligence_universities.pdf`. Read this article. What do you think the future of education would look like with the advent of such tools? Include 1-2 paragraphs.

11 When you're done

When you're all done you should have a working program that draws your picture. Make sure that your program is properly commented:

- You should have comments at the very beginning of the file stating your name, course, assignment number and the date.

- Each function should have an appropriate docstring

- Other miscellaneous comments to make things clear

In addition, make sure that you've used good *style*. This includes:

- Following naming conventions, e.g. all variables and functions should be lowercase.

- Using good variable names.

- Proper use of whitespace, including indenting and use of blank lines to separate chunks of code that belong together.

12 What to submit

For this assignment you should submit three things AT THE SAME TIME (see below):

1. Your `assign2.py` file. (See "Grading" below for the list of required functions.)

2. A copy of your final picture. The easiest way is to do a screen capture:

    On mac:
    Hold command+shift+4 and then while holding these, hit the spacebar (on a mac). If you then click on the window where your picture is drawn, an image file will be saved on your desktop entitled "Screen shot...". You can double-click on this file and then print it. You may have to click "scale" in the print options to get it to fit on one page. It won't print it color if you print in the lab, but this will give us most of the details

    On windows:
    There are plenty of resources online about how to do a screen capture. One option is:
    `http://graphicssoft.about.com/cs/general/ht/winscreenshot.htm`

3. Your response to the ethics readings as a pdf file.

Submitting multiple files: Go to the online course submission and select "assign2". When prompted to select your file, you need to select *all* files. On Mac, you can do this by first clicking one file, then, holding down the command key, click the second file, etc. On Windows, first select the first file, then, holding down the control key, click the second file, etc.

Grading

		points
`triangle`		
	equilateral	1
	correct x, y	1
	correct direction	1
`polygon`		
	correct shape	2
	varying sizes	1
	x, y, direction	1
`add_circles`		
	correct number of circles	1
	covers entire area	1
	random locations	2
	random sizes	1
`generate_picture`		
	background color	1
	proper fills	1
	6+ fish/ships	2
	6+ rocks/planets	2
`rectangle`		
	correct shape	1
	correct coloring	3
Comments, style		3
Ethics prompt		1
extra credit		1
total		26 (+1)