Project 1: Making Shapes with Turtle

CS 5001 Intensive Foundations of Computer Science

1. **Goals:**
   - Establish your coding environment
   - Explore Python turtle as a mean of learning simple computational thinking
   - Get practice creating basic Python applications

2. **In Recitation (Instructions for what's to be done in recitation and for the recitation ICE):**

   Most labs will have an associated recitation section. The main purpose of the recitation is to make sure you are prepared to complete the weekly assignment. Recitation 1 will be about making sure you have the basic tools needed to do the work and to review this week's assignment.

   Recitation Goals:

   - Get your Visual Code installation up and running
   - Become familiar with project creation and text editing
   - Familiarize yourself with Python turtle

   Objective 1 : Getting your IDE

   You should be familiar with the three ways to compile a program: cloud, command line, and IDE, but for recitation I'd like you to focus on getting your IDE ready and you making sure you are comfortable with project setup, creating an application, and saving your code files.

   Later you are welcome to use any IDE that you'd like, but we'll be using Visual Studio code for demonstrations and recitation work. It can be downloaded here: https://code.visualstudio.com/

   You should have this downloaded and installed on your computer prior to the first recitation. If you'd like to work ahead, you can also explore some online resources to learn more about how it works: https://code.visualstudio.com/docs

   For your ICE submission for objective 1, take a screenshot of your Visual Studio installation. I want to see that you have it ready for the next class.

Objective 2 : Text editing

Text editors are the workhorse programs for writing code. You don't want fancy fonts or WYSIWYG layouts, you just want to see lines of text, preferably with syntax highlighting, which means that special words in a language are highlighted to make it easier to read the code. There are many editors to choose from.

For class exercises I will be using Visual Studio Code, which is a free download for Windows, Mac, and Linux. VS Code also has an integrated Unix terminal, which we will use for executing our programs. For now, go ahead and open up VS Code.

Steps

1. On the Welcome view, select Add New Workspace...
2. Create a working directory for your first lab and project. You probably want to have a main CS5001 folder and within that have a folder for each project.
3. On the Welcome view, select New File.
4. At the top of the file type `print("You are smart")`
5. Select Save file (or cmd-S or ctrl-S) and save the file as smart.py
6. Under the Terminal menu, select New Terminal. This should create a sub-panel at the bottom of your screen with a prompt.
7. At the prompt in the terminal, type *ls* and then hit return. You should see the file smart.py
8. At the prompt in the terminal, type python smart.py. This will execute the code in the file smart.py, which should display the string you told it to print in the terminal window.

Congratulations, you just ran your first Python program. (Python humor) (Links to an external site.)

Note that the above command works only if the current working directory is the one that contains the file smart.py. So, one thing you need to do when working on a project is to ensure that your Terminal is in the right directory. Think of it as making sure VS Code and Terminal agree about where the program is located in your file system. You can always use ls to see the contents of the current directory.

In general, you run Python programs by typing

python3 <filename>

For your information

Here are some other text editors you should know about:

- Atom - a simple and free code editor that is available for MacOS, Windows, and Linux.
- Sublime Text - a non-free code editor (but free if you can put up with ads) available for MacOS and Windows.
- Notepad++ - a free and open source text editor for Windows.
- Visual Studio Code a free and open source code editor developed by Microsoft, available for MacOS, Windows, and Linux.
- Bracket is a free and open source code editor developed by Adobe, available for MacOS, Windows, and Linux.
- nano/pico - nano (or pico on some systems) is a very simple text editor that works in a terminal. All of the commands use the control key and are listed at the bottom of the screen. To open up a file, just type the name of the editor followed by the name of the file. If the file does not exist nano creates the file.
- BBEdit - BBEdit is an upscale version of TextWrangler designed for writing code and html (web pages).
- emacs - emacs is a text editor that has been used heavily by computer scientists for almost 40 years. It has many powerful capabilities, and the key commands in emacs are used in many other places (like TextWrangler). emacs is the battery-powered swiss-army knife with optional nuclear reactor of the editor world (emacs humor (Links to an external site.)).
  If you open emacs, the opening screen tells you how to access a tutorial. The most important thing to know is how to save and get out. To save, hold the control key down and type x then s (C-x C-s). It will ask you for a filename (and give you a default if you opened a file) and then save it. To exit, hold down the control key and type x then c (C-x C-c).
- JEdit - JEdit is very similar to TextWrangler, and you can download it for free and run it on any operating system. Many students use it since it runs on Windows. It is slightly more complex than TextWrangler.
- XCode - XCode is a powerful integrated development environment for Mac OSX, but you can also use it just for its editor, which is pretty decent.

Note: Terminal is a very useful tool to use and we'll explore its use further later.

For your ICE submission for objective 2, take a screenshot showing you can run a "hello world" application in Visual Studio

Objective 3 : Drawing with Python turtle

Python is a simple yet powerful interpreted language for making computers do stuff. Many people have written modules for Python that do some sophisticated things. One of those modules is called turtle and implements turtle graphics (e.g. forward, left, right, pen down, and pen up).

We can make the turtle move forwards, backwards, turn left, turn right, and pick up or put down the pen using the following commands.

- forward(x) - move forward x pixels
- backward(x) - move backward x pixels
- left(a) - turn left a degrees
- right(a) - turn right a degrees
- up() - pick up the pen (don't draw when the turtle moves)
- down() - put the pen down (draw when the turtle moves)

More complete documentation is available on the Python documentation site (Links to an external site.).

Writing Your First Turtle Program

1. **Create a new code file**
   In VS Code, select New File from the file menu.

2. **Always put the proper header at the top of each code file.**
   Type your name, a date, and the course and project in a comment at the top of the file. Every Python file you ever write should start with these three lines. In addition, put in a short comment that says what the code in the file will do.

3. **Import the turtle package**
   On the first line of your code file, put the instruction:

   ```
   from turtle import *
   ```

   That will import all of the turtle commands (i.e. it will give you the ability to direct the turtle to draw for you.)

4. **Write code to draw a line**
   After the import statement, call the function that makes the turtle go forward. Give it an argument of 100.

   ```
   forward(100)
   ```

The turtle should move forward, drawing a line 100 pixels long.

5. **Write the code to wait for user input when done drawing**
   On the last line, put the instruction:

   ```
   mainloop()
   ```

   That will keep the turtle window open until you close it.

6. **Save your file**
   Save your code file in your working directory. Call the file square.py because we will use it to draw a square in the next step. Notice that the filename ends in with the .py extension, which indicates the file contains Python code.

7. **Test your code**
   Run the program from the terminal using python square.py. What does it do?

8. **Have the turtle draw a square**
   Using the right and forward turtle functions (commands), make your code draw a square instead of just one line. Write one command on each line, and put them in between the import and mainloop commands. Try running your code again to see if it does what you expect.

   ```
   forward(100)
   right(90)
   forward(100)
   right(90)
   forward(100)
   right(90)
   forward(100)
   right(90)
   ```

For your ICE submission for objective 3, take a screenshot showing your square

3. **Lab Assignment Instructions (instructions for what's to be submitted to Canvas) :**

The purpose of this project is to give you a chance to break down simple problems (such as drawing a logo) into precise steps that solve the problem (draw the logo). In computer science speak, you will be developing algorithms. In this case you will write your algorithms as sequences of Python turtle commands that make the turtle draw complicated shapes for you.

Setup Your Workspace

Create a new working directory for project 1. For this project you will be creating at least two files: shapeOne.py and shapeTwo.py. Each file will generate one picture using turtle graphics. If you do one or more extensions, then if they require separate files please name them extension1.py, extension2.py, and so on as needed. Extensions don't need to be in separate files (for example adding color or other drawing characteristics to your shapes does not require a separate file).

Tasks

1. **Design your first shape**
   Invent a shape, such as a cross, or an L or a simple chair. Do this on a piece of paper or white board. You may want to use graph paper so you can count the length of the sides of the shape. Here is a template you can print out if you would like: graph paper pdf.

   You will need to use turtle commands to draw the shape, so keep that in mind. Any shape is possible, but some shapes are more difficult to code than others. You can make disconnected elements by using the up and down turtle commands so you can move the turtle without drawing. Take some time to understand what turtle can do before just diving in.

2. **Code your shape**
   Create a file called shapeOne.py in your project1 directory. It should follow the same basic format as the python file we made in lab and class: the first and last lines in shapeOne.py are the same (from turtle import * / mainloop()). Between those lines, put the turtle commands to draw the shape you just invented. Make sure to include a comment block at the top with your name, the date, and a comment about why you are writing this application. Run the program (i.e. type python3 shapeOne.py in the terminal).

   Take a screenshot of your image. This is your first image for your report.

3. **Design a second shape**
   Design a second shape in the same file. It doesn't need to be any more or less complex than your first shape, just different. Get creative. See if you can figure out how to place it somewhere else in the image.

4. **Add a scale factor to your second shape**
   Create a file called shapeTwo.py in your project1 directory. Use the same format as for the first shape, however, after the import statement, assign to the variable scale the value 1.0. (*scale = 1.0*) When you code your second shape, whenever you call a function that moves the turtle (forward or backward), multiply the argument being passed to the function by the variable scale.

   For example, if you have the instruction forward(100), then you should write forward(100*scale)

Run your code and capture a screenshot of your shape with scale having the value 1.0. Then change the value of scale to something else (e.g. 1.5 or 0.5) and capture a screenshot of the shape with the different scale value. Make sure the two shapes are different sizes.

The two pictures of shape two are the second and third required pictures needed for your report.

4. Extensions

Projects are your opportunity to learn by doing. They are also an opportunity to explore and try out new things. Rather than list out every activity for a project, the defined part of each project will be about 85% of the assignment. You are free to stop there and hand in your work. If you do all of the required tasks and do them well, you will earn a B/B+.

To earn a higher grade, you can undertake one or more extensions. The difficulty and quality of the extension or extensions will determine your final grade for the assignment. Extensions are your opportunity to customize the assignment by doing something of interest to you. Each week we will suggest some things to try, but you are free to choose your own.

A common question is "how much is an extension worth?" The answer to that question depends on the extension, how well it is done, and how far you pushed it. As teachers, we prefer to see one significant extension, done well, where you explore a topic in some depth. But doing 2-3 simpler extensions is also fine, if that's what you want to do. Choose extensions that you find interesting and challenging.

Only a short explanation of extensions will be provided in future assignments.

The following are a few suggestions on things you can do as extensions to this assignment. You are free to choose other extensions, and we are happy to discuss ideas with you. Be sure to save and include pictures of your extensions in your report.

- Add color to your shape
- Create more shapes than what was requested by using more built in turtle commands
- Make a really complex shape
- Draw out your own personal logo

5. Report

Each week you will need to submit a report along with your code. Your report is primarily intended to demonstrate that you have completed all the tasks and to show you executed your application. The report is worth 4 points of your final grade and each lab will tell you what section it needs to contain. You may use a word processor of your choice, but please submit your report as a pdf with the name **report01.pdf** along with your code. This week I'd like the following sections:

1. **Problem Description**
   Create a brief statement about this assignment including goals you were meant to accomplish. This should be no more than 4-5 sentences.
2. **Required Task Elements**
   Include all requested screen shots. Each one should have a brief statement about what it is showing.
3. **Extensions**
   Describe any extensions you did for this project so that you get credit for them. Include images or outputs that demonstrate your extension works.
4. **Reflection**
   Reflect back on the assignment. Write a brief description (1-3 sentences) of what you learned. Think about the answer to this question in terms of the stated purpose of the project. What are some specific things you had to learn or discover in order to complete the project?
5. **Acknowledgements**
   Provide a list of people you worked with, including TAs, professors, and other students in the course. List also any resources that you used. This includes textbooks, lecture notes, python documentation, library documentation, websites, or forums.
6. **Grading Statement** (optional) Using the rubric, what grade do you think you deserve and why?

6. Submission

You should be submitting at the least:

- shapeOne.py
- shapeTwo.py
- Report01.pdf

Submit your assignment in the Canvas under Lab 1 : Making Shapes with Turtle

When you submit, double-check the following.

- ☐ Does your code compile and run?
- ☐ Is your name and an appropriate header at the top of each Python file?
- ☐ Does your report have all requested sections completed?

☐ Is your report a pdf document?

7. Rubric

	Possible	Given
Main Objectives		
Simple functionality with shapeOne.py	6	0
Simple functionality with shapeTwo.py	6	0
Scaling added to shapeTwo.py	6	0
Misc		
Report	4	0
Code Quality (correct indentation, comment blocks, variable naming, etc)	4	0
Not included in total possible:		
Extensions (Not calculated without report)	4	0
Creative or went above and beyond	4	0
Code does not compile	-30	0
Late penalty	-6	0
Not implemented as requested	-30	0
TOTAL POINTS POSSIBLE out of 30	26	0