# T3: Boustrophedon Turtles and Functions

* Assignment T3 should be done with a partner.
* To begin, go to “File” and Select “Make a Copy...”
* Move your copy of the document to the folder for your section.
* Rename the file to **T3: Boustrophedon Turtles - usernames** (replace usernames with your usernames). To do this, click the label in the top left corner of your browser.

## Learning Objectives:

* Explore functions in Python
* Understand the difference between arguments, parameters, and variables
* Learn how to customize functions using arguments
* Understand return values of functions

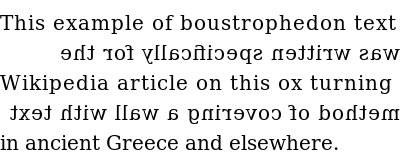
#### 

First, discuss with your team and assign yourselves roles. Try to pick the role you’ve had the least experience in.

|  |  |
| --- | --- |
| **Driver[[1]](#footnote-0):** |  |
| **Navigator[[2]](#footnote-1):** |  |
| **Quality Control[[3]](#footnote-2) (if the class is odd numbered):** |  |

## What is Boustrophedon?

Boustrophedon writing (ancient Greek: βουστροφηδόν) means turning like oxen do when plowing. It is a kind of bi-directional text which is mostly seen in ancient manuscripts.



In this assignment, your will design two different algorithms which use the Python Turtle library to draw a rectangle and fill that rectangle using a boustrophedon pattern. These algorithms must substantially differ. i.e. just changing the colors or the text would not be sufficient.

## Boustrophedon pattern using a Python Turtle

The Python Turtle screen has a default horizontal width of 683 pixels and a default vertical height of 576 pixels. The default setting for (0, 0) is in the center of the screen.



The following video uses a pensize size of 20. It first draws a purple square with sides of 500 pixels around the center. It then fills using a green horizontal boustrophedon pattern, and finishes using black text. Double-click the image below, and then double-click the video to play.



And here is an implementation of a second algorithm:



## Useful Examples of Functions and Turtle Methods

Here are some examples of code which is organized into functions:

* [turtle-multicolor-squares.py](https://drive.google.com/open?id=0B0J8Yj0B6KRSWW9RR0JXVXY5Nk0)
* [turtle-spiral-input.py](https://drive.google.com/open?id=0B0J8Yj0B6KRSeFUyaTJrZEVaRHM)
* [Turtle-functions-house.py](https://drive.google.com/open?id=0B0J8Yj0B6KRSWlRycy1ZQXJzejg)
  + [Bricks.gif](https://drive.google.com/open?id=0B0J8Yj0B6KRSRzY0ZWxhU1RwNVU)
  + [deck.gif](https://drive.google.com/open?id=0B0J8Yj0B6KRSZFNnYVVpblNfcVk)
  + [Lighthouse.gif](https://drive.google.com/open?id=0B0J8Yj0B6KRSZnlMc2tSX0MxeEk)

## The instructions

Before you start, read. There are hints and requirements below that will impact your design of the algorithms.

|  |  |
| --- | --- |
| Design two different algorithms which could be used to draw a large square and fill it using non-overlapping, just touching boustrophedon patterns as shown in the videos. Describe your algorithms on the right in plain English (not code, yet).  To design the first algorithm, try the design on a piece of paper, moving and turning your pen or phone as if it was a turtle. As you are considering the orientation and position of the turtle, you are considering changes of the turtle's state. Note that you may design for algorithmic convenience, you need not have the exact number of turns shown in the videos.  A bit of arithmetic will help with the following requirements:   * You must use at least one for loop. * You must use a pen size between 20 and 30. * The sides of your large outer squares must be 500 or larger. * Your boustrophedon patterns **must be** non-overlapping and just touching. No gaps! | 1.a. Algorithm 1:  1.b. Algorithm 2: |
| Decide which of these two designs you will implement. Explain the key reasons for your choice. | 2. |
| Identify the parts of the algorithms that would make useful functions.  What would you name the functions, and what inputs would those functions need? | 3. |
| Your code must include a main() function. See the examples. We will talk about why later, but for now, take a guess as to why a main() function is necessary. | 4. |
| Use docstrings to help clearly explain what each function is designed to do.  Why do we include docstrings to functions? | 5. |

Additional helpful hints and requirements to the assignment:

1. The highest level of your program (i.e., no indenting) should **only** contain the following:
   1. the header
   2. any import statements
   3. function definitions
   4. a call to the main() function
2. All of your own function definitions should come before the def main(): function definition AND the call to the main() function. In other words, the last lines of your code should be:  
    def main ():  
    # your code inside of main  
     
    main()
3. All of your own function definitions should be at the highest level of your program (i.e., no indenting). Though it is possible to do, functions should really not be DEFINED inside of other functions (they can be CALLED inside other functions though).
4. Using the typical program header for the class.
5. Once you have your implementation working, pretty it up by adding color, comments in the code, and text on the canvas.

|  |  |
| --- | --- |
| Describe the challenges of this programming assignment. Did you struggle with getting the screen coordinates to work? Did the programming go smoothly after you selected the algorithms? Explain. | 6. |

## Submission Instructions

1. (Submitter) Save your initials code as **T3\_boustrophedon\_usernames.py**. Replace *usernames* with your Berea usernames. For example, the TA Bianca Marrero and my document would be named **T3\_boustrophedon\_heggens\_marrerob.py.   
   NOTE:** Incorrect filenames will automatically reduce your grade by 1 point. Fortunately, the format is always the same no matter what the assignment.
2. (Submitter) Upload the Python file to Moodle by the due date listed on the course website: <https://trello.com/b/w7bIrLoV/>.
3. (All Other Partners) Open up Wordpad. Create a new text document (.txt) and include all members names in it.
4. (All Other Partners) Save the document as ***T3\_usernames.txt***. Replace *usernames* with your Berea usernames. For example, the TA Bianca Marrero and my document would be named **T3\_heggens\_marrerob.txt.**
5. (All Other Partners) Upload the document to Moodle by the due date listed on the course website: <https://trello.com/b/w7bIrLoV/>.

1. The driver will be doing the majority of the typing in PyCharm. Your job is to solve the problem given to you by the Navigator. [↑](#footnote-ref-0)
2. The navigator will be giving directions to the driver, and helping the driver catch syntax and logic errors as he or she creates the code. The navigator should keep track of time and make sure progress is being made. [↑](#footnote-ref-1)
3. The quality control specialist will ensure rules are being followed, both in the code (suggesting places to add comments, watching for misspellings, etc.) and in this document (making sure the questions are being answered at the right times, checking for typos, etc.) In a group of two, everyone is responsible for quality control. [↑](#footnote-ref-2)