CS100 Fall 2014

Name _____

CPADS Programming Activity II – Due 10/24

"Super Turtle!"

The goal of this section of the course is to introduce fundamental programming constructs using a simple scripting language, Python. This approach will allow us to focus on *programming* rather than *syntax*, i.e. formulating a procedural solution. To accomplish this task we may write both *console* programs that process text files, as well as *turtle graphics* programs where we draw graphics in an "Etch-a-Sketch" fashion.

1. Slow and Steady.

We will now write our first substantial Python program. For this program we will use a turtle graphics library known as *Swampy* (http://www.greenteapress.com/thinkpython/swampy/). In the turtle graphics world, we move a virtual turtle around the screen using only a few simple commands (hence *planning* will be important). Additionally, the turtle can pick *up* or put *down* the pen. The commands are:

`fd(t, length)` – moves turtle `t` forward `length` units
`bk(t, length)` – moves turtle `t` backward `length` units
`lt(t, angle)` – turns turtle `t angle` degrees to the left
`rt(t, angle)` – turns turtle `t angle` degrees to the right
`pd(t)` – starts drawing for turtle `t` (pen down)
`pu(t)` – stops drawing for turtle `t` (pen up)

- Open PyCharm (**Menu->Programming->PyCharm**).

- In PyCharm, create a new project by selecting **File->New Project…**
  When the **Create New Dialog** windows pops up:
  - name your project **Activity2**
  - set the location for **Activity2** to be the same **CS100** directory you created during Activity 1
  - ensure that the Interpreter is set to Python 3.4.0
  - click **ok** to create the new project

- Create a new Python file in your project
  - Right-click on the **Activity2** in the left hand side of your IDE and select **New->Python File**
  - Name your new Python file **`rightang.py`**
  - Your new file should open up in the editor panel of your IDE

Name _____

- Type the code below into your **rightang.py** file. **NOTE:** programming languages are almost always case-sensitive. Be careful and type the code EXACTLY as it is shown below.

```
# Load TurtleWorld functions
from TurtleWorld import *

def main():
      # Create TurtleWorld object
      world = TurtleWorld()

      # Create Turtle object
      turtle = Turtle()

      # Draw graphics
      fd(turtle, 100)
      rt(turtle, 90)
      fd(turtle, 100)
      rt(turtle, 90)

      # Press enter to exit
      key = input('Press enter to exit')
      world.destroy()

main()
```

- Save your **rightang.py** program

- Run your program by selecting **Run->Run**... and then selecting **rightang** in the pop-up box that appears.

  Sketch the output produced in the TurtleWorld graphics window. When done, click in the console window of PyCharm and press **Enter** to close the TurtleWorld window.

To understand a bit more about this program, **from TurtleWorld import *** tells Python to import the entire TurtleWorld library which is needed for this program. The next line **world = TurtleWorld()** is used to create a turtle graphics window. The following line **turtle = Turtle()** creates a new turtle and assigns it to the variable **turtle**. The next four lines then issue movement commands to **turtle** to perform the drawing. Finally, the last two lines simply keeps the turtle graphics window open until we press enter (in the IDLE window) to close it. Lines that begin with the # character are comments and can be used to document your program. Commented lines are not executed and can contain anything.

Name _____

2. It's not *magic*.

The program you entered above has several *magic numbers* that will make the program both difficult to read and maintain. In this part, you'll make a few changes to your previous program to remove the *magic numbers*.

- Make a copy of your **`rightang.py`** program
  - Right-click on the **`rightang.py`** file in the left panel and select **Copy**
  - Right-click on the **`rightang.py`** file again and select **Paste**
  - In the Copy dialog box that appears, give the new file the name **square.py**
  - Click **ok**
  - You will now have two files in your **Activity2** project. Double-click on the **square.py** file to ensure that you're editing you new file for the remainder of this part.

- Modify **square.py** to remove the magic numbers and instead store those values in variables with descriptive names **length** and **angle**.
  - Create two assignment statements before the **fd** and **rt** drawing commands. The first assignment statement should assign a value to a variable named **length**. The second assignment statement should assign a value to a variable named **angle**. Note that these variables MUST be declared and assigned a value BEFORE they can be used.
  - Add a comment above your newly created variables to document their intended use

- Replace the magic numbers in the **fd** and **rt** drawing commands with the appropriate variables. Note that you can use each variable more than once.

- Add four more drawing commands to have the turtle draw a square and end up back where it started and facing right.

- Save your **`square.py`** program

- Run your program by selecting **`Run->Run`**... and then selecting **`square`** in the pop-up box that appears. Show the instructor your program executing.

Name _____

3. If it's broke, fix it.

- Create a new Python file in your project
  - Right-click on the **Activity2** in the left hand side of your IDE and select **New->Python File**
  - Name your new Python file **`triangle.py`**
  - Your new file should open up in the editor panel of your IDE

- Type the code below into your **`triangle.py`** file. **NOTE:** programming languages are almost always case-sensitive. Be careful and type the code EXACTLY as it is shown below.

```
# Load TurtleWorld functions
from TurtleWorld import *

def main():
      a = TurtleWorld()
      x = Turtle()
      t = 100
      fd(x, t)
      z = 60
      rt(x, z)
      fd(x, z)
      z = 100
      rt(x, z)
      fd(x, t)
      rt(x, z)

      # Press enter to exit
      key = input('Press enter to exit')
      world.destroy()

main()
```

The program you entered above is intended to draw an **equilateral triangle**. What output does the program produce?

What good programming practices were not followed and what corrections need to be made to make the code more readable and produce the proper output?

Make these changes and show your instructor the corrected version.

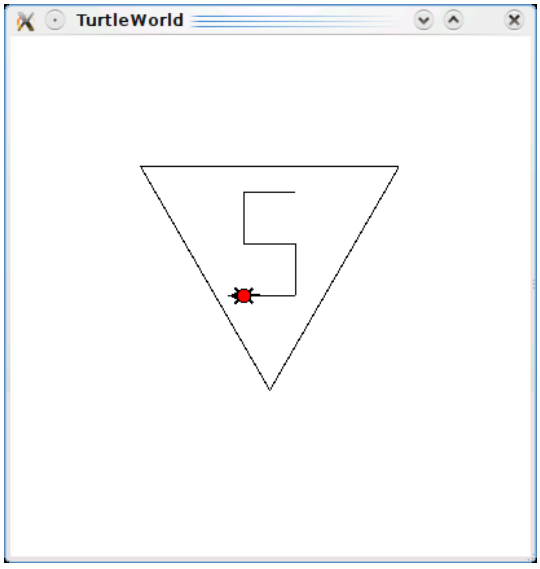
Name _____

4. It's a bird, it's a plane, it's *super turtle!*

Now it is time to make things a bit more complicated. You should do some planning with pencil and paper *before* you start typing any code.

- Create a new Python file in your project
  - Right-click on the **Activity2** in the left hand side of your IDE and select **New->Python File**
  - Name your new Python file **`superturtle.py`**
  - Your new file should open up in the editor panel of your IDE

- Using your program from Part #3 as a template, write a program to produce output shown below. The requirements are as follows:
  - Define a variable named **length** that represents the length of the side of an **equilateral triangle**. A good starting value for **length** is 100.
  - The figure must scale based on *the single* **length** variable.
    - You will probably want to create additional variables that are computed based on **length**. For example, you might create a second variable called **slength** that is computed as one-fifth the size of **length**.
  - The figure should be centered in the screen with the S roughly centered in the triangle. **Hint:** The turtle starts in the center of the *screen* so think about how to reposition the starting point *before* drawing the triangle.
  - Your program should not have any *magic numbers* and should be commented appropriately.



- Save your **`superturtle.py`** program

- Run your program by selecting **`Run->Run`**... and then selecting **`superturtle`** in the pop-up box that appears.

Name _____

- Once your program output looks like the figure shown above, try changing the value of **length** from **100** to **200**. Run your program again. Does your output look correct? If not, you will need to fix how you're scaling the various features of the Super Turtle logo.

- When you're ready to submit your program, print out and STAPLE a copy of your `superturtle.py` file to this activity.

- Submit your source file through Marmoset.
  - Open a web browser (e.g. Google Chrome) and enter the following URL (continue to the website if it brings up a certificate error page)

    https://cs.ycp.edu/marmoset/

  - Enter your login information which you should have received in an e-mail (you probably should change your password to match your YCP account)
  - Select **CS100: Computer Science Practice and Design Studio**
  - Select the **submit** link under **web submission** for **program01**
  - Click **Choose File…** , navigate to your program directory and select your **superturtle.py** file (do not worry about the instructions for jar and zip files).
  - Click **Submit project!**