CS100 Fall 2014

Name ________________________________

CPADS Programming Activity III – Due 10/31

"Now I Know My S,T,O,P's!"

One very useful and powerful programming construct that makes code both more readable, modular, and flexible is *functions.* Functions are typically used when the same non-trivial operation is done frequently but with different values. A function allows us to define the generic behavior, i.e. *encapsulation*, based on a set of *parameters* and then *call* the function whenever we wish with particular *arguments* (i.e. specific values for the parameters). This technique also helps make our code more readable by replacing a complex sequence of statements with a single meaningfully named function call. Furthermore, if we write functions well, they can be used by other people who only need to know *what* the function does (including what values they need to provide and what type of value they can expect to receive back) without consideration of *how* the computations are performed. Likewise, we can take advantage of functions other people have written without being concerned with the actual underlying code. For example, the square root function (**`sqrt()`** in Python) finds the square root of a number for us without us knowing how the computation is performed.

In Python, functions are usually defined at the top of the source code file using the following syntax

```
def func_name(param1, param2, …):
      statements
      return return_value
```

where *`func_name`* is the name of the function and (*`param1, param2, …`*) is the parameter list of variables that will be used *within* the function. The function *body* consists of one or more **`statements`** composed of valid Python code (including declaring *local* variables within the function). Those **`statements`** can use the parameters as well as any locally defined variables to perform a desired computation. If desired, a function can return a *single* value use a **`return`** statement at the end of the function *body*. Note that in Python, the body of the function is denoted by *indentation* (unlike C/Java which denotes the function body with {}).

Once we have defined a function, we may use it anywhere we wish, including in the main program and/or in other functions, by making a function *call* at the location we wish to invoke the function's behavior using the syntax

```
var = func_name(arg1, arg2, …)
```

where *`var`* is a variable in which to store the return value (if the function returns a value) and *`arg1, arg2, …`* are *expressions* (including values and variables) whose *value* will be *passed* to the function (i.e. used to set the corresponding *parameters* when the function executes). Hence we must have the same number of arguments in the function *call* as there are parameters in the function declaration (in the same order), but they do not need to have the same names. We must be careful to pass arguments of appropriate types to ensure the function performs correctly (later after learning about decisions we will be able to test the arguments prior to passing them to the function).

Name ________________________________

1. You make the call.

The first exercise is similar to the first program from the previous programming activity, but this time using a function to draw a right angle.

- 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 **Activity3**
  - set the location for **Activity3** 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 **Activity3** in the left hand side of your IDE and select **New->Python File**
  - Name your new Python file `rightangfunc.py`
  - Your new file should open up in the editor panel of your IDE

```
# Load TurtleWorld functions
from TurtleWorld import *

# Right angle function
def right_ang(t,size):
      fd(t,size)
      rt(t,90)
      fd(t,size)
      rt(t,90)

# Main program function
def main():
      # Create TurtleWorld object
      world = TurtleWorld()

      # Create Turtle object
      turtle = Turtle()
      turtle.delay = 0.01

      # Define variables
      length = 100

      # Draw graphics
      right_ang(turtle,length)

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

# Call main program
main()
```

Name ________________________________

- Save your `rightangfunc.py` program
- Run your program by selecting **`Run->Run`**… and then selecting `rightangfunc` in the pop-up box that appears. Verify that the code works as you expect before proceeding (i.e. it draws a right angle).
- Now, let's make a small change. After the *call* to `main()`, (i.e. at the very end of the program), add a line of code that calls the `right_ang()` function. Pass the arguments `turtle` and `length` in this function call. What is the output of this program? Explain any error messages that appear.

- Move the call to `right_ang()` that you added at the end of the program to inside the `main()` function immediately after the existing call to `right_ang()`. Modify the second argument in this new call to make it twice the value of length, i.e. **2*length**.
- Run your program again by selecting **`Run->Run`**… and then selecting `rightangfunc` in the pop-up box that appears.

  Sketch the output produced in the turtle graphics window.

Name ________________________________

2. "Give Me a P!"

```
# Load TurtleWorld functions
from TurtleWorld import *

# Right angle function
def right_ang(t,size):
      fd(t,size)
      rt(t,90)
      fd(t,size)
      rt(t,90)

# Draw P function
def draw_P(t,height):
      # Put pen down before drawing the letter
      pd(t)

# Main program function
def main():
      # Create TurtleWorld object
      world = TurtleWorld()

      # Create Turtle object
      turtle = Turtle()
      turtle.delay = 0.01

      # Define variables
      length = 100

      # Draw graphics
      draw_P(turtle,length)

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

# Call main program
main()
```

Using the skeleton code above (based on the previous program), write the function **`draw_P()`** that takes two parameters, **`t`** representing the turtle used to draw and **`height`** representing the height of the letter such that:

- The function draws a block **P** where the **width is one-half of the height** and assumes the turtle cursor *starts* in the *upper-left* corner and is facing *right.*
- **The turtle cursor MUST *end* in the *upper-right* corner and be facing *right when the function finishes.***
- *All* drawing **MUST** be done using the **`right_ang()`** function (which ***CANNOT*** be modified). In the **`draw_P()`** function, the turtle can be repositioned without drawing, i.e. when the pen is up, using any of the turtle graphics drawing commands.

Save the program as **drawP.py** and show the instructor your output.

CS100 Fall 2014

Name ________________________________

3. STOP POP TOPS!

The greatest advantage to using functions is the ability to perform the same operations at different points in our programs. Use your **drawP.py** program as a reference and the skeleton code below to complete part #3. See the directions on the next page for details on your task.

```
# Load TurtleWorld functions
from TurtleWorld import *

# Right angle function
def right_ang(t,size):
      fd(t,size)
      rt(t,90)
      fd(t,size)
      rt(t,90)

# TODO: Draw P
def draw_P(t,height):
      pd(t)

# TODO: Draw O
def draw_O(t,height):
      pd(t)

# TODO: Draw Space
def draw_Space(t,height):
      pu(t)

# TODO: Other drawing functions

# Main program function
def main():
      # Create TurtleWorld object
      world = TurtleWorld()

      # Create Turtle object
      turtle = Turtle()
      turtle.delay = 0.01

      # Define variables
      letter_height = 75

      # TODO: Draw graphics

      # Create Inspector Turtle to show the center of the TurtleWorld window
      inspector = Turtle()

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

# Call main program
main()
```

CS100 Fall 2014

Name ________________________________

You must write several functions to produce an output similar the screenshots on the next page. Specifically, you will need functions to draw the letters **P**, **O**, **S**, and **T**. Additionally, you will need functions to insert white space as necessary.

AS YOU WRITE EACH FUNCTION, add a temporary function call inside the `main()` function to *test* your new function for correct operation using `turtle` and `letter_height` as arguments. Refer to the example screenshots on the next page for the shape of each letter.

A more detailed description of your task is below.

- Create a new Python file in your project
  - Right-click on the **Activity3** in the left hand side of your IDE and select **New->Python File**
  - Name your new Python file `stoppoptops.py`
  - Your new file should open up in the editor panel of your IDE

- Write the skeleton code from the previous page into your `stoppoptops.py` file

- Write the function `draw_O()` that takes two parameters, `t` representing the turtle used to draw the letter and `height` representing the height of the letter such that:
  - The function draws a block **O** where the **height is twice the width** and assumes the turtle cursor *starts* in the *upper-left* corner and is facing *right.*
  - <u>**The turtle cursor MUST *end* in the *upper-right* corner and be facing *right.***</u>
  - Consider using the `right_ang()` function when appropriate, but you may draw some segments of the **O** without the `right_ang()` function.

- Write the function `draw_Space()`such that:
  - It moves the turtle forward 1/5 the height of the letters without drawing

- Call the `draw_P()`, `draw_O()`, and `draw_Space()` functions in the `main()` function to create the word **POP** in the <u>**center**</u> of the turtle graphics window.
  - The center of the **O** in **POP** should be centered in the turtle graphics window. A turtle called `inspector` is created in the skeleton code to mark the center of the window.

- Write the functions `draw_S()` and `draw_T()` following the same guidelines listed above for the `draw_O()` function. Again, assume the turtle starts at the upper-left corner of the letter and must end at the upper-right corner.

- Call the various drawing functions to write the word **STOP** above the word **POP** in the turtle graphics window.
  - Before writing the word **STOP** reposition the turtle at the start of the **S** in **STOP**
  - Make the spacing between *lines* 1/4 the height of the letters.
  - The word **STOP** must also be centered horizontally in the turtle graphics window.

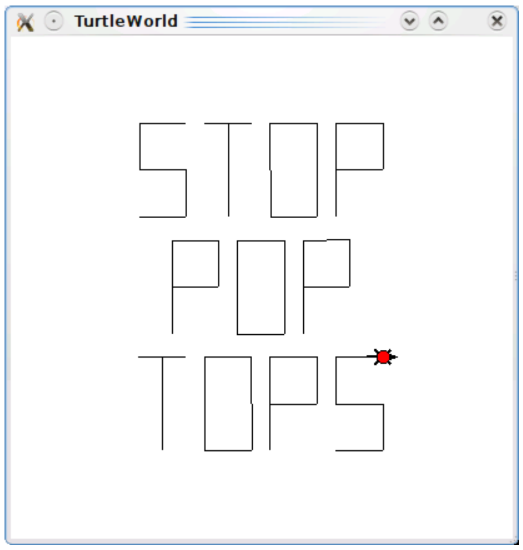
Name ___________________________________

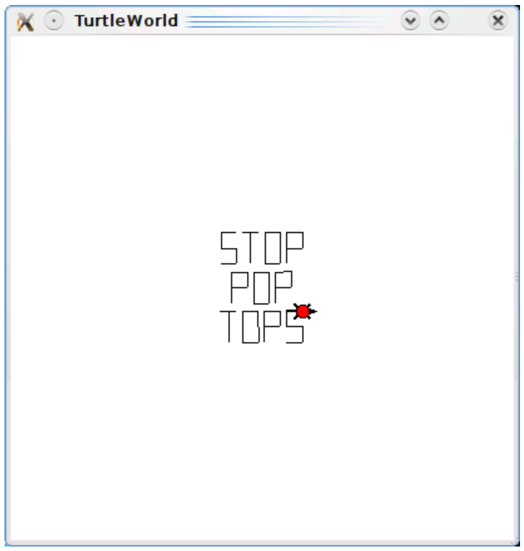
- Call the various drawing functions to write the word **TOPS** below the word **POP** in the turtle graphics window.
  - Before writing the word **TOPS** reposition the turtle at the start of the **T** in **TOPS**
  - Make the spacing between *lines* 1/4 the height of the letters.
  - The word **TOPS** must also be centered horizontally in the turtle graphics window.

- Once your program output looks like the screenshot shown below, try changing the value of the `letter_height` variable in `main()` from **75** to **25**. Run your program again. Does your output look correct? If not, you will need to fix how you're scaling the letters.

- When you're ready to submit your program:
  - Print out and STAPLE a copy of your `stoppoptops.py` file to this activity.
  - Submit your source file through Marmoset (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 **program02**
    - Click **Choose File…** , navigate to your program directory and select your `stoppoptops.py` file (do not worry about the instructions for jar and zip files).
    - Click **Submit project!**

Hints:
- **THINK BEFORE YOU CODE.** A little thoughtful planning with paper and pencil will pay large dividends when it comes time to write code.
- The screenshot on the left uses a `letter_height` value of 75 while the one on the right uses a `letter_height` value of 25 without any other modifications to the program.


TurtleWorld
STOP
POP
TOPS


TurtleWorld
STOP
POP
TOPS