CS 1301 – Spring 2009

Homework 2 – Myro/Python Introduction

Due: Friday, Jan 16th at 6 PM

Out of 100 points

Files to submit: **1. face.py**
2. tipCalculator.py

For Help:

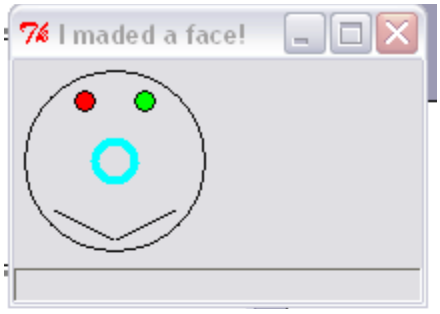
- TA Helpdesk – Schedule posted on class website.
- Email TAs

Notes:

- **Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).**
- **Do not wait until the last minute** to do this assignment in case you run into problems.
- If you find a significant error in the homework assignment, please let a TA know immediately.

Myro Intro

So you're sitting there, looking at your Python window, and you find it to be, well, impersonal. So why don't we add a little personality to it? How about…a face!



Hey! That looks much friendlier. And you can do it all with python! Specifically, that infamous myro thing you've installed. Myro contains a multitude of commands for programming your robot. See the documentation here:

http://wiki.roboteducation.org/Myro_Reference_Manual

For this assignment, we will be focusing on the graphics aspect of myro, so you won't need your robot. Both the link given above, as well as the PDF for the Myro Graphics

Reference will be rather helpful.

Part 1 – Making the Face (50 Points)

Your mission, should you choose to accept it (and we recommend that you do), is to make a face in Python. You need to create a program that initializes a graphics window, and draws the face. When you are done, save your program as "face.py". **Make sure that your filename matches this exactly, or you will lose points!** Don't forget to include any necessary import statements needed to use myro functions.

The face should contain:

- A Head
- 2 Eyes
- A Nose
- A mouth
- At least 2 different colors

Try to keep things in a general face alignment here (Sorry to all you budding Picassos out there), but the shapes can be whatever you want. Be creative!

Some Useful Functions

Python is full of all kinds of useful functions, used from everything to user input to number manipulation to creating directories and reading files. For this homework, you will need to use functions that gather information from the user, cast values to the correct type, and print things out to user's screen.

First, let's look at the two functions for gathering user input, **raw_input()** and **input()**. **raw_input()** takes a response from the user and saves it **as a string.** It accepts all types of input, but keep it mind it will always be saved as a string. The parameter for this function (the stuff inside the parenthesis) is an optional prompt that you can display to the user so they know what kind of input you want. **input()** works almost the same, except that input is used **only** to take in numbers. You can pass it either ints or floats, and it will save the values appropriately.

Sample:

```
>>> x = raw_input("Age? ")
Age? 6
>>> type(x)
<type 'str'>
>>> y = input("Age? ")
Age? 4
```

```
>>> type(y)
<type 'int'>
>>> z = input("Age? ")
Age? 14.5
>>> type(z)
<type 'float'>
```

Now that we've got some input from the user, we need to be able to work with it. In the first example, we used raw_input and the user gave us a number, but the value is stored as a string. You can't do math with a string. However, we have a way to turn that string into either an integer or a floating point number. This is called **casting**, and you use functions like **int()**, **float(),** and **str()**, depending on what type of data you need.

Sample:

```
>>> x = "15.6" # Here x is a string (note the quotes)
>>> x = float(x)
>>> type(x)
<type 'float'>

>>> x = 1111 # Here x is a interger
>>> x = str(x)
>>> type(x)
<type 'str'>
```

Casting (especially to floats) comes in very handy for a lot of mathematical operations, and being able to cast to strings helps out in print statements.

Speaking of printing, it's important to be able to display things to the user. Now, when I say printing, I don't mean connecting to your HP and actually printing out pages using ink; rather, I mean showing something to the screen, typically a result from some computations. You can print most anything, as long as it's formatted properly.

Sample Code:

```
print "Hello"    # Printing a simple string
print 57.6    # Printing a number
var = 21
#Three ways to print the same thing
print "Your age is:" , var
# This just prints the variable on the end
print "Your age is: %i" % var
# This uses placeholders (i for ints, f for floats, s for strings)
print "Your age is: " + str(var)
#This uses string concatenation (all terms must be strings)
```

Resultant Output:
>>>
Hello
57.6
Your age is: 21
Your age is: 21
Your age is: 21

You can also use the % operator (string formatting) if you need to trim decimal places off to create neater output.

Sample Code:
x = 23.4567
print "Number: %.2f" % x

Output:
>>>
Number: 23.46

Part 2 – It's Math Time! (50 points)

So you've been up all night playing Rock Band™ with your friends. Well, rocking out hardcore always makes one tired, so you decide to go grab some food. It being nearly 3 am, the only place is the great institution of Waffle House. However, you're spent from a night of constant awesomeness, and realize you just won't be able to compute the appropriate tip amount for the waitress in your head. For some reason, though, the part of your brain that writes Python scripts is super focused and ready to go, so you decide to whip out your laptop and write a Python program to handle the tip calculations for you. (Yes, you brought your laptop to Waffle Hours with you. You are a Tech student, after all.)

You program will also need to be able to handle tax.

-Write a script that prompts the user for the bill amount and the percent that they want to leave for the tip. **Do not include the dollar sign or percent sign when entering the numbers!**

-Calculate the tax (**always 8%**), the tip (percent of bill amount, **excluding** tax), and the total bill.

-Remember to divide the percentages by 100 when multiplying. Here, it is important to remember how Python handles division.

-You are expected to round the your tip to the next dollar by use of the math.ceil() function. Basically, math.ceil() always rounds the number up to the next integer. **Do not forget to import math when using functions from the math module!**

For example:

```
>>> math.ceil(4.51)
5.0
>>> math.ceil(6.7)
7.0
>>> math.ceil(3.2)
4.0
```

-Round the tax to the nearest cent by use of string formatting. Remember, you're working with money, so how are dollar amounts usually presented?

Sample output:

```
>>>
How much is the bill before tax and tip? 7.80
What percent tip do you want to leave? 15
Tax is $0.62
Tip is $2.00
Total is $10.42
```

Save your script in a file called "tipCalculator.py". **Again, save your file with this exact name, or you will lose points!**

Turning it in

Once you're done, please submit face.py and tipCalculator.py to T-Square. The assignment is due Friday, Jan 16th before 6pm. If you're late, but turn it in before 6pm on Monday, you will lose 10%. We will not accept submissions after Monday.

Grading Rubric

Part 1 – Face – 50 points

- File named correctly (face.py) – 5 points
- Creates a graphics window – 10
- Draws a head – 6
- Draws 2 eyes – 6
- Draws a nose – 6
- Draws a mouth – 6
- Uses at least 2 different colors - 6
- The facial features are properly aligned – 5
- Total for part 1: 50 points

Part 2 – Tip Calculator – 50 points

File named correctly (tipCalculator.py) – 5 points
Gets input from user – 5 points
Input is cast correctly – 5 points
Tax is calculated correctly – 5 points
Tip is calculated correctly – 5 points
Total is calculated correctly – 5 points
Uses math.ceil() as needed – 5 points
Outputs are correct – 15 points (5 each for tax, tip, and total)
Total for part 2: 50 points

For a grand total of 100 possible points.
You can earn up to 5 points bonus [discretion of the TAs] for extra creativity/general awesomeness, for a possible total of 105/100.

Written By: Melody Nailor, Spring 2009