Computer Science 111 Spring 2014

Lab 2: Arithmetic

This lab lets you practice using Python statements for doing input, output, and arithmetic. Sit in front of a computer, log in, and start up IDLE in interactive mode.

1 Python Syntax Errors!

A **syntax error** is like a typo in your program: mismatched parentheses, mistyped variable names or keywords, misuse of semicolons and colons (about which more in a few weeks) and so forth. No matter what programming language you use, the program won't run if there is a single syntax error.

> **Tip:** Pay attention to how *exactly* a Python statement is written, and try to get your syntax right the first time – adopting this simple habit can save you hours of homework time over the semester.

Python is perfect at finding syntax errors, but not great in diagnosing what the problem is. Type the (error-filled) statements below, one by one. I have numbered them with comments for reference. Read the error message you get for each line and compare them to the notes below.

```
>>> class = "Computer Science 11"          #1
>>> first-name = "Monty"                   #2
>>> primt ()                               #3
```

1. The word `class` is a keyword, so is not allowed as a variable name. Python thinks you are misusing the keyword. There are 29 Python keywords (you can google them): don't use them for variable names.

   Otherwise here are the rules for choosing a variable name: it must start with a letter of the alphabet (lower or upper case ok, but avoid upper case for now); then it can have any combination of letters, numbers, or underscores. Python is case-sensitive, so the variable **strawBerry** is different from **strawberry**. For now, you should avoid starting a variable name with an underscore or with a capital letter because those variables tend to have special meanings in Python.

2. You can't have hyphens in variable names. Python thinks the hypen minus sign (an operator), and complains because you can't have operators on the left side of an =.

3. A misspelling of `print`. Python doesn't recognize the word `primt`.

Now get creative. Some more syntax errors are listed below. For each one, try to write a statement that deliberately creates the error. You have 5 minutes. Go!

1. **NameError** You use a variable in a way that requires a value but it hasn't been assigned a value.

2. **TypeError** you have the wrong type in your expression.

3. **ZeroDivisionError** can you guess?

More about print statements. Here are some variations on ways to use print statements.

```
>>>  m = 9
>>>  d = 13
>>>  y = 2013
>>>  print (m, d, y)    #print three values each separated by space
>>>  print (m, d, y, sep='/', end=" ")   #separated by slash,  no newline at end
>>> x = 10 / 3
>>> print (x)   # notice all the decimal places in this float.
```

You can use a `format` function to specify how you want a number to look when printed. The function is of the form

```
format ( value,  string)
```

, where *value* is some type of numeric (integer or float) value, and the *string* specifies how it should look. Here is a simplified table of string elements (you can look up the fancier things on the web if you like).

element	allowed values	meaning
align	`< > =`	justify left, right, center
width	int	min field width
comma	,	use commas in the output
precision	. int	number of decimal places
type	`f e d`	float, scientific notation (with exponent), decimal integer

Try these examples to see what the format function does.

```
x = 1234.56789
print ("x= ",  format (x , "3.2e"))
print ('The integer part is',  format (int(x) , "10d"))
print ("Total price: ",  format (x,  "=12,.2f"))
```

Formats are easier to understand if you read them right-to-left:

1. The first format says prints the value of x using scientific notation (e); use 2 decimal places; and use a minimum field width of 3. If the number won't fit into the minimum field width, the field grows so the whole number can be printed.

2. Print it as an integer (d), using 10 spaces for the field width. The number will be right-justified within that field. Note that `x`, which is a `float` has to be converted to an int to avoid a syntax error. Try it without the conversion and see what happens.

3. Print it as a float with two decimal places, and insert commas in the int part. The number is centered in a field of total width 12.

2 Lab 2 Assignment

Enough playing around. Now it is time to write some real code!

- Open a browser and point it to ...
  `www.cs.amherst.edu/~sfkaplan/courses/2014/spring/COSC-111/projects/lab2.py`

- Right-click and choose Save As to download `lab2.py`. Ok to save it on your desktop, but you probably want to move it to your U drive for permanent storage. Remember that items left on the college desktops get erased overnight.

- Open the file in IDLE: select **File: Open** and browse to find the file. Now you are working in compile mode in the File window.
- To run the program, select **Run: Run Module** from the top menu. You could also type FN-(F5) by holding down the Function key and the F5 key at the same time. When the program runs, you are the user and must respond to all prompts. See how that works?
- You can save your program by typing COMMAND-S on a Mac or CTRL-S on a Windows platform. **You have to save it before running it, every time you make a change.**
- Now go back to the File window and edit the file, following the instructions in comments. ***Get in the habit of re-saving and re-running after typing just a few lines (like 4 or less)!*** Seriously. Otherwise you you will spend all your time tracking down a ton of syntax errors. It's faster to work incrementally and fix them as you go.
- When you have each part finished, run the program to check that it works.
- When you are finished, point your browser to ...
  `http://app.cs.amherst.edu/Submit`
  Use the Submit program to turn it your working Python program by 9:00 MONDAY MORNING.
- Quit IDLE and Log Out before leaving the lab!