CSCI 111, Lab 1
Geoffrey Matthews

Due date: Midnight, Tuesday, September 13, on Canvas. No late work accepted.

File names: Names of files and variables, when specified, must be EXACTLY as specified. This includes simple mistakes such as capitalization.

Individual work: All work must be your own. Do not share code with anyone other than the instructor and teaching assistants. This includes looking over shoulders at screens with the code open. You may discuss ideas, algorithms, approaches, *etc.* with other students but NEVER actual code.

Setup: Log in to a computer in the lab and launch Python's IDLE from the menu. The interpreter window will have a title bar of IDLE Shell 3.10.0, where the version number may be different, but should start with 3.

Experiments with the shell: Enter expressions into the shell. Some of the responses will make sense, some won't yet, and some will be errors. After you get a feel for this, try to anticipate what the response will be before you see it. Some things to try:

```
128
3.14169
1
001
4 / 3
4 // 3
13 // 3
13 % 3
4 / 0
4 = 3
4 == 3
'hello'
'hello' + 3
'hello' * 3
```

The shell is in a REPL, a "read-eval-print-loop". It reads your expression, evaluates it, and prints the result.

Variables: Frequently it will help us to give names to things. For example, when solving the quadratic equation

$$ax^2 + bx + c = 0$$

It helps to give the name *discriminant* to the quantity

$$d = b^2 - 4ac$$

The name d is a new name, nowhere in the statement of the problem, that makes it easy to talk about something complicated. For example, we can simply say that if the discriminant is positive, there are two solutions, if the discriminant is zero there is only one solution, and if the discriminant is negative there are no real solutions. Also, we can simplify the presentation of the solutions

$$x = \frac{-b \pm \sqrt{d}}{2a}$$

We can give names to any object we want in Python. Such names are called *variables*. We use a single equals sign, `=` for this operation, which is called *assignment*. We *assign* a value to a variable. Here are some examples:

```
x = 99
y = 'hello'
z = 'hello' * 3
```

Note that there are some rules for variables. Use the shell to figure out which of the following can be used as variables, and which cannot.

```
grade
student
class
one
two
```

```
NowIsTheWinterOfOurDiscontent
fubar
foo_bar
f'bar
fub@r
```

Printing: If you are running a program NOT in the shell nothing is printed unless you want it to be. For this there is the `print` function. When you use the `print` function in the shell, it doesn't appear to do anything!

```
print('hello')
print(greeting)
greeting = 'hello'
print(greeting)
age = input('Enter your age:')
print(age)
print(age + 1)
print('Your age is', age)
name = input('Enter your name:')
print('Hello', name, ', you are', age, 'years old.')
```

Make sure you understand when print is required to get output. Also note that some of the commas in the last expression are inside strings, and some are part of the print statement. Also note that `print` is more versatile, since you can concatenate several things into a single printout.

Input: We can use `print` to get output from the program, but how can we get input into a program? Try the following:

```
s = input(‘Type something! ‘)
s
```

You may have noticed that after the first line was entered, the interpreter stopped running until you typed something. The `input` statement makes your program wait for the user to type some input.

Math: To do much math you'll need to import the math library.

```
pi
import math
math.pi
math.sqrt(2)
math.sqrt(-1)
math.sqrt(2)**2
dir(math)
help(math)
help(math.sqrt)
```

Types: Every object in Python has a *type*. Numbers are obviously different kinds of things from words or strings. You can do arithmetic on numbers, but (generally) not on strings. Python will tell you the type of something just by asking:

```
type(33)
type(3.14159)
type('hello')
type(math.pi)
type(math.sqrt)
```

Notice that Python uses the word class instead of type, but they are essentially the same and for the most part using the word type is more common.

Notice also that even a function, like `math.sqrt` has a type.

Comments: Finally, it is nice to have a way of telling Python to just ignore something. Anything we want Python to ignore is called a *comment* and is introduced with the # sign. See what happens when you enter these into the shell:

```
# 99
# What the heck?
# 99 + 'hello'
```

The editor: Running programs in the shell is good for small tasks and makes a good replacement for a calculator. However, if we're going to write long programs we need to write the program out and save it in a file. For this we use the IDLE editor.

- In the menu bar of the IDLE shell, click on File→New.
- A new window will open up with the title Untitled.
- Use File→Save to save your file. Make a new folder in your home directory, or your Box directory, called `CSCI111Labs`
- Make a new folder in `CSCI111Labs` called `lab01`
- Finally, save the file as `birthyear.py`
- Enter the following into the editor window, but use your name instead of mine:

```
# birthyear.py
# Geoffrey Matthews
# CSCI 111, Fall 2022
# input/output demo

name = input('Enter your name:')
age = input('Enter your age:')
birthyear = 2022 - age
print('Hello', name, ', you were born in', birthyear, '.')
```

This simple program follows the general order of a great many programs:

1. Input
2. Processing
3. Output

The first part is a comment block identifying the program, it's author, and a bit about its purpose.

All programs you hand in should begin with a similar comment block!

Run it: When you think you've got everything entered correctly, click Run→Run Module in the menu. An alternative is simply to press the function key F5

- Answer the prompts for name and age.
- What happened? There is an error in the program, and Python does its best to tell you what went wrong where. Mine looks like something like this:

```
Enter your name: Geoff
Enter your age: 71
Traceback (most recent call last):
  File ".../birthyear.py", line 6, in <module>
```

```
    birthyear = 2022 - age
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

- Python tells you that the two operands for - were of type 'int' and 'str' and that these types are not supported by -

Debugging: The process of finding errors in your program and fixing them is called debugging.

- Now we know what's wrong with the program, so it's time to fix it. In this case, we have read in our age, in my case, 71, as a string. In generall, all input will be input as string. But we want to convert this string to an integer. Fortunately, there is a simple way to convert something to an int. Try these in the shell:

```
type(99)
type('99')
int('99')
type(int('99'))
```

- Using this knowledge you should be able to fix program `birthyear.py` Fix it before you turn it in.

Now that you know a bit about programming, you should be able to write and debug the following programs on your own.

poem.py Write a program that has no input, but prints a short poem for output. Ideally this should be one you wrote yourself, but it's not necessary. Here is one of mine:

```
There once was a hacker named Fly
Who tried to rob Sundar Pichai.
    He trained his pet poodle
    To break into Google
But got caught when she fell for the guy.
```

quadratic.py Write a program that prompts for the a, b, and c of the quadratic formula and prints the two solutions (which can be found in these notes). If there's only one solution, it will print it twice. If there are no solutions then it can just crash with an error. Here is the behavior of my solution on some test cases:

```
Enter a: 1
Enter b: 4
Enter c: 1
The solutions are -3.732050807568877 and -0.2679491924311228
```

```
Enter a: 3
Enter b: -5
```

```
Enter c: 2
The solutions are 0.6666666666666666 and 1.0
```

```
Enter a: 1
Enter b: -2
Enter c: 1
The solutions are 1.0 and 1.0
```

```
Enter a: 1
Enter b: 1
Enter c: 1
Traceback (most recent call last):
  File "....quadratic.py", line 5, in <module>
    d = math.sqrt(b**2 - 4*a*c)
ValueError: math domain error
```

Upon completion: Zip your `lab01` folder with the three programs, `birthyear.py`, `poem.py`, and `quadratic.py` and turn in the zipped file to Canvas.

- Don't forget to include an identifying comment block at the beginning of each program.
- Be sure to logout when you leave the lab.