**A Brief Introduction to Computer Programming**

All of the programs that you use on computers were written by programmers. Strictly speaking, a computer only understands 0s and 1s and humans don't do so well communicating in 0s and 1s. Instead, many computer programming languages have been invented that are easier for humans to use, such as Python and C, the two languages you'll use in this class. These languages have strict rules, much like English or Spanish, but unlike regular human languages, these languages are completely unambiguous. This means that if you give a valid set of instructions following the proper rules (syntax), then these instructions have one clear meaning that the computer will execute. This is good and it's bad. It's good because the computer doesn't have a mind on its own. It won't ignore your instructions and do whatever is on its mind, like a human might, when given instructions. It's bad, because if you make a mistake in your directions (which we humans do frequently), the computer doesn't have any common sense to correct it. So, you have to be 100% correct with the instructions you give AND the order you give them in. In this class, we'll focus on learning the syntax of both Python and C, to express computer programs that solve relatively simple problems for us.

C is a compiled language. This means that for us to get a computer to execute our instructions in C, we must first write an entire program in C and save it in a file. All programs written in C are stored in .c files. For example, we might write a program and store it in the file hello.c. A program called a compiler, which is essentially a translator, is then run on the file hello.c. The compiler has two jobs: to make sure that the syntax, or rules of the C language, were followed in the C file and to translate that file to 0s and 1s that the computer will understand. This latter file is known as an executable file and typically has the file extension .exe. It's this file that can be executed so the user can see the results of the program.

Python is an interpreted language. This means that instead of directly translating a whole program from Python to 0s and 1s, that once a program is written, it gets interpreted. This means that small portions of it get translated from Python to 0s and 1s that the computer understands as the program is running. For our purposes, we'll notice the difference between Python and C in that we'll be able to run individual lines of Python code in the interpreter to see our results immediately, but for C, we'll have to complete our whole program, compile it successfully and then run that executable. This begs the question: what happens if we attempt to interpret a line of Python that is syntactically incorrect? The interpreter will tell us immediately.

**Running the Python Interpreter**

Once you've downloaded the Python interpreter and editor, IDLE, you are ready to use it. Just open it up and you'll be presented with a command prompt:

>>

You can type any python statement at this prompt and after typing enter, the result will be displayed. You can continue typing in statements and each subsequent result will be displayed immediately. If you make an error in using the rules of Python, the error will be displayed when you hit enter, instead of the results of running your command.

**Printing in Python**

The first command that is usually taught in any programming language is a command to print out a desired message. In python, the print function is used to do so. Here is an example of the print function:

print("Hello World.")

In general, the syntax of the print statement is as follows:

1) The word print.

2) Have a pair of matching open and closing parentheses right after the print.

3) Inside the parentheses, put opening and closing double quotes. Inside these double quotes, put your message, so long as it doesn't contain double quotes.

Any item in between a set of double quotes is called a string literal. A string is a sequence of characters and a literal is anything that doesn't change.

Try it yourself. Open the IDLE editor and type in a print command of your choice.

Incidentally, instead of using double quotes, one can use single quotes to delineate a string literal as well. Note that if we use double quotes to mark the beginning and end of a string, then we can put a single quote inside the string. Similarly, if we use single quotes to mark the beginning and ending of the string, we can put double quotes in the string. Try these examples:

print("It's a lovely day.")

print('Sam said,"Hello."')

We will later cover what to do if we want to print out both a single and double quote in the same print statement.

**Using a Variable in Python**

You can think of a variable as a box that can be used to store a particular value. The value stored in the box can change over the course of a program, which is why it's called a variable. In other programming languages you have to declare a variable and give it a type, but in Python you don't. You just have to use the variable and based on context, the Python interpreter will figure out what type of information you would like to store in that variable. It is the programmer's job to pick names for each variable she creates. For example, we can use a variable named radius and set it to 10 as follows:

radius = 10

This creates a box with the label radius with a 10 in it. The equal sign in Python (and most programming languages) is different than the equal sign in math. In Python, the equal sign is the assignment statement. It tells the computer to figure out the value of the right hand side at that snapshot in time and it changes the value of the variable on the left of the equal sign to that value. In this instance, radius is changed to 10. Another way to change radius to 10 is as follows:

radius = 4 + 2\*3

You'll notice a couple things here: \* is used for multiplication and arithmetic expressions in Python use order of operations similar to what you learned in math class. Basically, this entire expression is evaluated first to get the value of 10, and then that value is stored in the variable radius.

Just creating a variable and storing one value in it isn't terribly useful. Here are a couple subsequent statements that might make our program more useful:

pi = 3.14

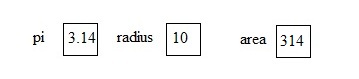
area = pi\*radius\*radius

If we execute these statements, in this order, right after we set radius to 10, area should be set to 314. One way to check this in our IDLE interpreter is to follow these statements by just typing the name of the variable we wish to check, as follows:

>> area

314.0

At this point in time, the picture in memory is as follows:

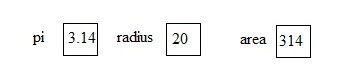


Now, after this statement, attempt the following two statements:

>> radius = 20

>> area

You might think that 1256 will print out for area, but that's not the case. Rather 314 prints out. The first statement above changes radius to 20, so our ensuing picture is:



As the picture clearly shows, area hasn't changed. Why? Because we didn't change it!!! We only changed radius. The = operator does literally what was previously described. It calculates the value on the left (in this case 20), then changes the variable on the right (radius) to that value. It does nothing more and nothing less.

If we want to change area to reflect this new value of radius, we have to again evaluate the following:

>> area = pi\*radius\*radius

>> area

Note About Printing Strings Literals and Variables in Python

It might be nice to output a statement of the following form:

The area of the circle is 314.0.

We can do this in a single print statement as follows:

>> print("The area of the circle is",area)

The rule of thumb is that if you want to print multiple things, just list each item you want printed inside of the print function call, separated with commas. The first item is a string literal, while the second item is a variable.

We can extend this idea as follows:

>> print("The area of a circle with radius",radius,"is",area)

In Python, when multiple components are printed in this manner, each item separated by a comma in the function call is printed with a space separating it in the output:

The area of a circle with radius 10 is 314.