Introduction to Python Programming

Andrew J. Pounds, Ph.D.

Exercise One: Screen Input and Output

In the following example, the program is going to request input from the screen.

```
#Simulate checking in for an appointment

first_name = input("Enter your first name: ")
last_name =  input("Enter your last name: ")
time = input("Enter time in hh:mm format: ")

print
print "Thank you " + first_name + " " + last_name
print
print "We have you checked in at " + time
```

When you run this program you have to enter your name in quotes because you are entering a string of text. Python handles many different types of data, but most program are written with three different types:

Strings are a collection of characters like “ABC Bail Bonding”. In the preceding string, there are actually sixteen characters as the spaces count as characters too.

Integers are positive and negative counting number like -10, 0, 8, etc.

Floats or “floating point” numbers are numbers with decimals. It should be recognized, however, that they are finite values. For example, the value of π as a float is 3.1415926535897. It does not go on forever -- it is truncated or rounded after a certain number of decimal places.

Do an experiment. Re-run the program and instead of entering the names and date in the expected format, just type 1, 2, and 3 respectively. You should see an error like this:

```
Traceback (most recent call last):
  File "checkin.py", line 8, in <module>
    print "Thank you " + first_name + " " + last_name
```

```
TypeError: cannot concatenate 'str' and 'int' objects
```

In the print statement the program is concatenating (putting together) and trying to print different data types. Since you just entered 1, 2, and 3 python treats them as integers. The “Thank you” in the print statement and the “ “ are strings. In general python wants homogenous data types in a print statement and this can be fixed via inline typecasting. Typecasting is when you convert one data type to another. For example, I could write the print statement as

```
print "Thank you " + str(first_name) + " " + str(last_name)
```

Where I am calling the “str” **function** to convert the argument to a string. This can be handy when you need to print things like strings either floats or integers together.