1.6 Reading User Input in Python

input statement

Python makes reading input from the user very easy. In particular, Python makes sure that there is always a prompt (a print) for the user to enter some information. Consider the following example entered

```
>>> name = input("What is your name?\n")
What is your name?
Simone
>>> print("Please to meet you ", name, ".", sep="")
Please to meet you Simone.
```

In this way, instead of the print always printing out the same name, it will print out whatever name the user entered. The key is that the input statement read in whatever the user entered, and then the assignment statement, with the equal sign, assigned this value to the variable name. Then, we were free to use name as we pleased, knowing that it stored the value the user entered.

Our previous programs that calculated the price of an item with tax and the area of a square, were limited because they always calculated the same price and area. Our program would be much more powerful if we allowed the user to enter the appropriate values so that our program could calculate the information THEY are interested in (as opposed to the same value every time.)

Before we look at the necessary edits to make these programs take in user input, one other note must be made about the input function. It always returns whatever it reads in from the user as a string. This means that if the user enters a number, such as 79, the input statement returns it as "79" instead. (This is a string that literally is the character '7' followed by the character '9', instead of the number 79.) Thus, we need some method to convert the string "79" to the number 79. The way this is done is through a function that changes its input to a different type. To turn a string into an integer, use the int function:

```
>>> age = int(input("How old are you, "+name+"?\n"))

How old are you, Simone?
22
>>> print("Your name is ",name,". You are ",age," years old.", sep="")
Your name is Simone. You are 22 years old.
```

In this example, we had to use the int function (on the first line) to convert the string the input function returned into an integer. Then, this was assigned to the variable age. Thus, age stores an integer and not a string.

In prompting Simone, you'll notice that plus signs, denoting string concatenation were used instead of commas, which we first used when we learned the print statement. The reason for this is that while the print statement takes in multiple items separated by commas (these are called parameters), the input statement only takes in a single string. Thus, we were forced to create a single string by using string concatenation. Since the variable name is a string, we were able to concatenate it with the rest of the message. Here is the error message we would have gotten had we attempted to pass the input function separate items separated by commas:

```
>>> age = int(input("How old are you, ",name,"?\n"))
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    age = int(input("How old are you, ",name,"?\n"))
TypeError: input expected at most 1 arguments, got 3
```

The last line of IDLE's output tells us what occurred. The input function expects 1 argument, or piece of information, but we gave it three pieces of information, since the commas are what separate pieces of information (arguments) given to a function.

In the ensuing print, we can give the print function multiple pieces of information separated by commas, and we see that age has indeed been stored as 22.

Other than reading in strings and integers, one may want to read in floating point numbers from the user. Since the input function always returns a string, that string must be converted to a floating point number instead of an integer. The function in Python that handles this conversion is the float function. Here is an example of reading in a floating point number:

```
gpa = float(input("What is your grade point average?\n"))
```