Lesson 4: Who Goes There?

In this lesson we will write a program that asks for your name and a password, and prints a secret message if you give the right password. While doing this we will learn:

1. What a function is and how to call it.
2. How to set the value of a variable to something that the user typed (input)

Let's begin creating the password program. Start IDLE. Create a new Editor window by selecting the “File” menu in the Python Shell window, then selecting “New Window”.

Type the following line in the new Editor window:

```
print “Hello”
```

Now let's save the file before we go any further. Let's call this program password.py. Select the “File” menu, then “Save”. In the “File name” box type **`password.py`**. Save the file in your **`MyPrograms`** directory. Please review “The Editor Window” on page 7 for instructions on how to save a file in your **`MyPrograms`** directory.

So far this is just the same as our hello.py program, but we will add more commands in a moment. Just for fun, press the F5 key to run this program. Python says:

```
Hello!
```

in the Python Shell window.

In the password.py window, put the cursor at the beginning of the first line, before any other characters, and press Enter. This creates a new blank line at the beginning of the file.

Type the following in the blank line you just created:

```
name = raw_input("Who are you? ")
```

Now find the line that looks like this:

```
print "Hello!"
```

and make it look like this:

```
print "Hello", name
```

Do you notice that there is an asterisk (*) before and after the file name in the title bar of the hello.py window? That means that we have made changes to the file but have not yet saved the changes. You can't run a new or changed program until it has first been saved.

Press Ctrl-S to save the file. “Ctrl-S” means press and hold down the Ctrl key (“control”), press the S key, then release the Ctrl key.

Now press the F5 key to run the program.

The Python Shell window should come to the front, and you will see a prompt from your very own program, asking you:

```
Who are you?
```

This is a question that we should all ask ourselves. What do you think will happen when you type your name and press Enter? Try it.

```
Hello David
>>>
```

FYI: If you typed your name but you didn't see anything, or it appeared in the wrong window, try clicking your mouse on the title bar of the Python Shell window first, so it gets the “focus”.

When you see the >>> prompt again in the Python Shell window you know that your program has finished running.

About Functions

Let's look at the **`password.py`** window again. Consider the command that you used to print a prompt and get input from the user:

```
name = raw_input("Who are you? ")
```

In the line above, the word `raw_input` is the name of a *function* that comes built-in to Python. Whenever you have a Python name followed immediately by a pair of parenthesis (), you are *calling a function*. A function is a piece of a program that you can run again and again in your own program to accomplish some specific task. In this case, `raw_input()` is a built-in function that gets input from the user. It prints a prompt and waits for the user to type something and press Enter. The function *returns* what the user typed.

The string inside the parenthesis is called the parameter to the function. It gives the function some information it needs to do its work. In this case the parameter is the prompt “Who are you?”. `raw_input()` will print its prompt parameter before it waits for input for the user.

After the user has typed something and pressed Enter, raw_input() *returns a value*. In this case the value is a string containing the text that the user typed. If you put the raw_input() command on the right-hand side of an = and there is a Python name on the left hand side, then the returned value is assigned to a variable.

Functions are a very useful concept. I will show you later in this book how to write your own functions.

More on the raw_input() Function

FYI: Why do they call it “raw_input”? Is there a *cooked* input? Yes, sort of. There is an `input()` function, but you don't want to use it very often. It takes the input and immediately executes it as a Python expression (command that results in a value.) That's probably not what you want, as you will often get errors.

Before we go on, let's practice calling the `raw_input()` function with different parameters. In the Python Shell window try the following:

```
>>> raw_input("What is your favorite color? ")
What is your favorite color? green
'green'
```

Notice that I put a space after the `?` and before the ending quote mark in the parameter. Otherwise it would look like this:

```
>>> raw_input("What is your favorite color?")
What is your favorite color?green
```

```
'green'
```

Now try a different parameter. When Python asks you how many lumps you want, just press Enter (because you don't want any lumps.)

```
>>> raw_input("How many lumps would you like? ")
How many lumps would you like?
''
```

(When it printed `''` that meant that raw_input() returned an *empty string*.)
Now try `raw_input()` without any prompt parameter. On the blank line after the `raw_input()` command type `test`.

```
>>> raw_input()
test
'test'
```

The Python Shell printed the return value for you. Notice that the returned string values have single quotes around them, like this: `'test'`. Python will accept double quotes or single quotes around strings, as long as you begin and end the string with the same kind of quote mark. That is one way that you can put quote marks inside of a string. Try this:

```
>>> print 'Did someone say "ice cream"?'
Did someone say "ice cream"?
```

Now let's go back to the **`password.py`** window and turn this program into something more interesting. (If you closed the **`password.py`** window, then open your **`password.py`** file again.)

Let's pretend that you have a secret that you want to share with everyone in the house but your sister (it's a good secret, a birthday present or something like that.) Let's assume that no one but you knows how to read and edit a Python program (that won't last for long if they read this book.) We'll put a secret message into this program that only someone with the right password can access.

Go to the end of the file and press Enter to insert a blank line. (Python doesn't mind blank lines.) Add the following lines to the end of the program:

```
message = "Sally is getting a bicycle for her birthday."
password = raw_input("What is the password? ")
if password == "birthday":
    print message
else:
    print "Wrong password."
raw_input("Press Enter to finish.")
```

Press Ctrl-S to save the program and F5 to run the program.

If you typed everything correctly, you should see something like this in the Python Shell window:

```
Who are you? David
```

```
Hello David
What is the password? birthday
Sally is getting a bicycle for her birthday.
Press Enter to finish.
>>>
```

Go back to the **password.py** window and press F5 again. If you use the wrong password it should look like this:

```
Who are you? Sally
Hello Sally
What is the password? horse
Wrong password.
Press Enter to finish.
>>>
```

If your program does not run correctly at first, there is no need to worry. This is a long enough program that there is a good chance you made a mistake in typing somewhere. Almost any small mistake will cause a problem that shows up when you press F5. I can't list all of the possible symptoms that you will see, but here are a couple of examples:

1. **A dialog pops up that says "There's an error in your program: invalid syntax."** Look in the program window and you will see something highlighted in red. The problem is somewhere at or before the place marked red - sometimes even on the line before. Check your typing in that area and try again.
2. **It prints a message that begins with "Traceback (most recent call last):".** Look at the very last line of the message, it will often give you a clue. If you misspelled a name in your program, it will say something like "NameError: name 'messag' is not defined".

For more help getting your program to run, see "Common Problems and Solutions" on page 157.

Congratulations on getting this far. You have learned some really useful building blocks. Perhaps this little password program has given you some ideas. However, it doesn't seem very useful to run a program with a password-protected message in it when the program window is open right there for anyone to see. Would you like to be able to have your Mom run this program, without her seeing the program code? I thought so. See "Appendix 2: How to Let Others Run Your Programs" in page 159.

Hiding the password while it is being typed

To avoid showing the password when it is typed, use the `getpass()` function from the `getpass` module (we'll explain modules later). Replace this line in **password.py**:

```
password = raw_input("What is the password? ")
```

with these two lines:

```
import getpass
password = getpass.getpass("What is the password? ")
```

If you run your program from the Python Shell, it prints the following warning:

```
Warning: Problem with getpass. Passwords may be echoed.
```

and prints the password anyway. But if you set up your program to be run from a shortcut using the instructions starting on page 159 the `getpass()` function will work correctly and not show the password.

FYI: Why is there an extra `raw_input()` call at the end of **`password.py`**, with the prompt “Press Enter to finish”? If you run the program from an icon on the desktop, the program window disappears as soon as it has finished running. The purpose of this line is to keep the program window visible until the user has had time to read what was printed.

Quiz 4

1. Change the secret message in **`password.py`**.
2. Change the password in **`password.py`**.
3. Change **`password.py`** so that it doesn't ask for your name and doesn't print it out.

After each of these changes, run the program by pressing F5. Fix the program if it doesn't do what you want.