Lesson 11. Interacting with the user

After reading lesson 11, you'll be able to

- Print values for the user
- Ask the user for input
- Store user input in variables and do operations with it

Many programs are written to do computations behind the scenes, but few of them are useful without some sort of input from a user. One main reason you'd want to write a program is to provide users with a certain experience; that experience relies on a back-and-forth between the user and the program.

Consider this

Find another person and have a conversation. What kinds of questions can you ask? What responses do you get? Can you build upon a specific response you get?

Answer:

- How are you?
- Good, looking forward to the weekend.
- Me too! Any weekend plans?
- Yes, we're going camping, then checking out the science museum. If we have time, maybe hit the beach and then going out for a nice dinner. You?
- Watching TV.

11.1. SHOWING OUTPUT

To get started with this lesson, recall that you can use the `print()` command to show the user values on the console in Python. You'll use `print` a lot from now on.

11.1.1. Printing expressions

You can put any expression inside the parentheses of `print()`, because all expressions evaluate to a value. For example, the float `3.1` has a value `3.1`, and the expression `3 * "a"` has a value `"aaa"`.

Listing 11.1 shows how to print the values of a few expressions to the user. You can have fairly complex expressions in the parentheses. The code in this listing prints out the following:

```
hello!
89.4
abcdef
ant
```

Listing 11.1. Printing expressions

```
print("hello!")                          1
print(3*2*(17-2.1))                      2
print("abc"+"def")                       3
word = "art"                             4
print(word.replace("r", "n"))            5
```

- *1* **A string**
- *2* **A mathematical expression**

- **5 Replaces "r" with "n"**

Notice that in every example in this listing, what you put in the parentheses isn't necessarily an object of type `str`. For example, `print(3*2* 17-2.1))` evaluates to an object of type `float`. The `print` command works with any type of object in the parentheses.

Quick check 11.1

Write each statement in a file in the editor and run the file. What do the following statements print, if anything? Type them in Spyder to check yourself:

1

```
print(13 - 1)
```

2

```
"nice"
```

3

```
a = "nice"
```

4

```
b = " is the new cool"
print(a.capitalize() + b)
```

11.1.2. Printing multiple objects

It's possible to place multiple objects in the parentheses after `print` and mix and match their types. If you want to put in different objects, separate each object by a comma. The Python interpreter automatically inserts a space between the values of your printed objects. If you don't want the extra space, you'll have to convert every one of your objects to strings, concatenate them together, and use this in the parentheses of `print`. Listing 11.2 shows an example. In the program, you want to divide one number by another and print the result. The code in this listing prints the following:

```
1 / 2 = 0.5
1/2=0.5
```

Notice that the first line that is printed has a space between every object, but the second line doesn't.

Listing 11.2. Printing multiple objects

```
a = 1                                          1
b = 2                                          1
c = a/b                                        2
print(a,"/",b,"=",c)                           3

add = str(a)+"/"+str(b)+"="+str(c)             4
print(add)                                     5
```

- **1 Initializes variables**
- **2 Calculates the division**
- **3 Uses commas to separate the integers (variables a, b, and c) and the strings ("/" and "=")**
- **4 Converts the integers to strings with (str) and then uses the + operator to concatenate them with the strings "/" and "="**
- **5 Prints the string**

Quick check 11.2

Convert each of the following points into a Python statement to create a program. After you're finished, run the program to see what's printed:

1

Make a variable named `sweet` with the string value `"cookies"`.

2

Make a variable named `savory` with the string value `"pickles"`.

Make a variable named `num` with the int value `100`.

4

Write a `print` statement that uses as many of the variables as you can to print `100 pickles and 100 cookies`.

5

Write a `print` statement that uses as many of the variables as you can to print `I choose the COOKIES!`

11.2. GETTING USER INPUT

The fun in creating programs comes when you can interact with the user. You want to use input from the user to guide calculations, computations, and operations.

11.2.1. Prompting the user

You can use the `input()` command to get input from the user. Suppose you want to ask the user to input their name. In the parentheses of `input()`, you put in a string object that represents the prompt for the user. For example, the line

```
input("What's your name? ")
```

will show the following text on the console and then wait for the user to type something:

```
What's your name?
```

Notice the extra space at the end of the prompt string. Figure 11.1 shows the difference between a string prompt with a space and without. You can see that whatever text the user will type in starts immediately after the end of the prompt string. A good rule is to leave a space as the last character in your prompt string so that the user can distinguish the prompt from their input.

Figure 11.1. How to prompt the user for input

Quick check 11.3

Write a line of code for each of the following:

1

Ask the user to tell you a secret.

2

Ask the user to tell you their favorite color.

3

Ask the user to enter any of the following: `#, $, %, &,` or `*`.

11.2.2. Reading the input

After prompting the user for input, you wait for the user to type something. If you're testing your program, you can take the role of the user and type in different things yourself. The user indicates that they're finished by hitting Enter. At that point, your program continues executing the line right after the one asking for input.

The code in the following listing shows a program that asks the user to input the city they live in. No matter what the user inputs, the program then always prints `I live in Boston.`

Listing 11.3. Where does the user live?

```
input("Where do you live? ")            1
print("I live in Boston.")              2
```

- **2 After the user hits Enter, this executes, and the program ends.**

Notice that the program isn't doing anything with the user input. This is an interactive program, but it's not particularly interesting or useful. More-complicated programs store the input from the user into a variable and then do operations on it.

11.2.3. Storing the input in a variable

Most programs act on user input. Anything a user types is converted into a string object. Because it's an object, you can bind it to a variable by assigning the input from the user to a variable. For example, `word_in = input("What is your fav word? ")` takes whatever the user inputs and stores it in the variable named `word_in`.

The following listing shows how to use the user input to print a more customized message. No matter what the user inputs, you make the first letter of their input capitalized, add an exclamation mark to the end of it, and then print the result along with a final message.

Listing 11.4. Storing the user input

```
user_place = input("Where do you live? ")          1
text = user_place.capitalize() + "!"               2
print(text)                                        3
print("I hear it's nice there!")                   3
```

- ***1* Gets user input and stores it in the variable user_place**
- ***2* Concatenates two strings: the user's input, capitalized, with an exclamation mark**
- ***3* Prints a customized message**

After you get the user's input as a string, you can do any operations on it that you're allowed to do on strings. For example, you can convert it to lowercase or uppercase, find indices of substrings, and check whether certain substrings are in the user's input.

Quick check 11.4

For each of the following, write lines of code to achieve the sample output:

1

Ask the user for the name of their favorite song. Then print the name of the song three times on separate lines.

2

Ask the user for a celebrity's first and last name. Then print the first name on one line and the last name on another line.

11.2.4. Converting the user input to a different type

Anything that the user types in is converted to a string object. This isn't convenient when you want to write programs that manipulate numbers.

Listing 11.5 shows a program that asks the user for a number and prints the square of that number. For example, if the user enters `5`, the program prints `25`.

You need to understand a few things about this program. If the user inputs something that's not an integer, the program will end immediately with an error, because Python doesn't know how to convert anything that's not a string whole number to an integer object. Run the program in listing 11.5 by typing in `a` or `2.1` for the user input; both will cause the program to crash and show an error.

When the user gives a valid number (any integer), recall that even though it looks like a number, everything the user inputs is a string. If the user types in `5`, Python see this as the string `"5"`. To work with the number, you must first convert the string into an integer by *casting* it—surround the string with parentheses and the type `int` before the string object.

Listing 11.5. Calculations with user input

```
user_input = input("Enter a number to find the square
num = int(user_input)
print(num*num)
```

- ***1* Gets user input and stores it**
- ***2* Converts the user's input to an integer**

- **3 Prints the square of the number. The first two lines can be merged into num = int(input("Enter a number to find the square of: ")).**

Quick check 11.5

Q1:

Modify the program in listing 11.5 so that the output printed to the console is a decimal number.

11.2.5. Asking for more input

You can write programs that ask for more than one input from the user. Listing 11.6 shows a program that asks the user for one number, then another, and prints the result of multiplying those numbers. But instead of only printing out the result, you're also printing helpful additional text that tells the user which operation you're doing and on which numbers. For example, if the user enters `4.1` and `2.2`, the program shows `4.1 * 2.2 = 9.02`.

Listing 11.6. Calculations with more than one user input

```
num1 = float(input("Enter a number: "))
num2 = float(input("Enter another number: "))
print(num1, "*", num2, "=", num1*num2)
```

- **1 Gets one number and converts it to a float**
- **2 Gets another number and converts it to a float**
- **3 Pretty-prints the multiplication by showing the two numbers you're multiplying and their result**

SUMMARY

In this lesson, my objective was to teach you how to show output and how to get input from the user. You learned that you can print multiple objects by using only one `print` statement and that Python automatically adds a space between each object.

You learned about using the `input()` command to wait for user input. The command converts anything that the user enters into a string object. If you want to work with numbers, you have to convert the input to an appropriate type yourself in the program code. Here are the major takeaways:

- `print` can be used on multiple objects in one go.
- You can ask the user for input as many times as you want. Each time, the program halts and waits for the user to enter something, and users indicate that they're done by pressing the Enter key.
- You can convert the user input into other types to do appropriate operations on it.

Let's see if you got this...

Q11.1

Write a program that asks the user for two numbers. Store these numbers in variables `b` and `e`. The program calculates and prints the power b^e with an appropriate message.

Q11.2

Write a program that asks the user's name and then age. Use appropriate variable names to store these variables. Calculate how old the user will be in 25 years. For example, if the user enters `Bob` and `10`, the program should print `Hi Bob! In 25 years you will be 35!`