2.15. Input

Our programs get more interesting if they don't do exactly the same thing every time they run. One way to make them more interesting is to get **input** from the user. Luckily, in Python there is a built-in function to accomplish this task. It is called `input`.

```
n = input("Please enter your name: ")
```

The input function allows the programmer to provide a **prompt string**. In the example above, it is "Please enter your name: ". When the function is evaluated, the prompt is shown (in the browser, look for a popup window). The user of the program can type some text and press `return`. When this happens the text that has been entered is returned from the `input` function, and in this case assigned to the variable `n`. Run this example a few times and try some different names in the input box that appears.

Save & Run   5/13/2021, 7:20:01 PM - 4 of 4   Show in CodeLens

```
n = input("Please enter your name: ")
print("Hello", n)

```

```
Hello Tom
```

Activity: 1 -- ActiveCode (ac2_16_1)

It is very important to note that the `input` function returns a string value. Even if you asked the user to enter their age, you would get back a string like `"17"`. It would be your job, as the programmer, to convert that string into an int or a float, using the `int` or `float` converter functions we saw earlier.

Note

We often use the word "input" (or, synonymously, argument) to refer to the values that are passed to any function. Do not confuse that with the `input` function, which asks the user of a program to type in a value. Like any function, `input` itself takes an input argument and produces an output. The input is a character string that is displayed as a prompt to the user. The output is whatever character string the user types.

This is analogous to the potential confusion of function "outputs" with the contents of the output window. Every function produces an output, which is a Python value. Only the print function puts things in the output window. Most functions take inputs, which are Python values. Only the input function invites users to type something.

Here is a program that turns a number of seconds into more human readable counts of hours, minutes, and seconds. A call to `input()` allows the user to enter the number of seconds. Then we convert that string to an integer. From there we use the division and modulus operators to compute the results.

Save & Run   4/17/2021, 1:56:55 PM - 2 of 2   Show in CodeLens

```
str_seconds = input("Please enter the number of seconds you wish to convert")
total_secs = int(str_seconds)

hours = total_secs // 3600
secs_still_remaining = total_secs % 3600
minutes =  secs_still_remaining // 60
secs_finally_remaining = secs_still_remaining  % 60

print("Hrs=", hours, "mins=", minutes, "secs=", secs_finally_remaining)

```

```
Hrs= 0 mins= 9 secs= 20
```

Activity: 2 -- ActiveCode (ac2_16_2)

The variable `str_seconds` will refer to the string that is entered by the user. As we said above, even though this string may be `7684`, it is still a string and not a number. To convert it to an integer, we use the `int` function. The result is referred to by `total_secs`. Now, each time you run the program, you can enter a new value for the number of seconds to be converted.

Check your understanding

data-16-1: What is printed when the following statements execute?

```
n = input("Please enter your age: ")
# user types in 18
print(type(n))
```

[✓ A. <class 'str'>]

B. <class 'int'>

C. <class 18>

D. 18

Check me   Compare me

✓ All input from users is read in as a string.

Activity: 3 -- Multiple Choice (question2_16_1)

You have attempted 4 of 3 activities on this page

✓ Completed. Well Done!