Lesson 6. Capstone project: your first Python program—convert hours to minutes

After reading lesson 6, you'll be able to

- Read your first programming problem
- Walk through two possible solutions
- Write your first Python program

Here are some of the main ideas you should be familiar with so far:

- Programs are made up of a sequence of statements.
- Some statements initialize variables.
- Some statements can be expressions to do calculations.
- Variables should be given descriptive and meaningful names, especially to help future programmers who might be looking at the code.
- Some calculations you've seen so far are addition, subtraction, multiplication, division, remainder, and power.
- You can convert an object to a different type.
- The `print` command can be used to show output to the console.
- You should write comments in the code to document what the code is doing.

The problem

The first programming task you'll see is to write a program in Python that converts minutes to hours. You'll start with a variable that contains the number of minutes. Your program will take that number, do some calculations, and print out the conversion to hours and minutes.

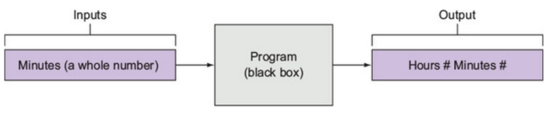
Your program should print the result in the following way. If the number of minutes is 121, the program should print this:

```
Hours
2
Minutes
1
```

6.1. THINK-CODE-TEST-DEBUG-REPEAT

Recall that before you begin to code, you should make sure to understand the problem. You can get the big picture by drawing your program as a black box. Figure 6.1 shows your program as a black box, any inputs, and any outputs you must generate.

Figure 6.1. The input to the program is any whole number representing minutes. The program does some calculations and prints out how many hours that is and any leftover minutes.



After you understand the inputs and outputs, come up with a few inputs and write down what you expect the outputs to be for each. Here are some other possible inputs for the number of minutes and their conversions to hours:

- "60 minutes" is converted to "1 hour and 0 minutes".
- "30 minutes" is converted to "0 hours and 30 minutes".

These input-output pairs are called *sample test cases*. You'll be able to use these inputs and expected outputs to test your program after you write it.

Quick check 6.1

What's the expected output given the following input for the number of minutes?

1

456

2

0

3

9999

6.2. DIVIDE YOUR TASK

Now that you understand what the problem is asking, you have to figure out whether you can break it into smaller tasks.

You need to have input to convert, so this can be one task. You're showing the user a result, and this can be another task. These two tasks are going to be easy to implement with at most a couple of lines of code.

Code to set up the input

To set up the input, you need to initialize a variable with a value. Your variable name should be descriptive, and the number of minutes should be an integer. For example,

```
minutes_to_convert = 123
```

Code to set up the output

To show the output to the user, the format required is as follows, where `<some number>` is calculated by your program:

```
Hours
<some number>
Minutes
<some number>
```

You show the user output by using the `print` command. Here's the code:

```
print("Hours")
print(hours_part)
print("Minutes")
print(minutes_part)
```

Here, `hours_part` and `minutes_part` are variables you'll calculate in your program.

Now the only thing left to do is to come up with a way to do the conversion from minutes to hours and minutes. This is going to be the most involved part of the overall task.

6.3. IMPLEMENT THE CONVERSION FORMULA

When you're dealing with time units, you know that 1 hour has 60 minutes. Your first instinct may be to divide the number of minutes you're given by 60. But the division gives you a decimal number: at a first pass, given 123 minutes, your result will be 2.05, not 2 hours and 3 minutes.

To do the conversion properly, you must divide the problem into two parts: find out the number of hours and then find out the number of minutes.

6.3.1. How many hours?

Recall that given 123 minutes, dividing 123/60 gives 2.05. Notice that the whole number part, 2, represents the number of hours.

Quick check 6.2

Divide the following numbers by 60 and determine the whole number part of the result. You can do the division in Spyder to check yourself:

1

2

0

3

777

Recall that in Python, you can convert one type to another type. For example, you can covert the integer 3 to a float by using `float(3)` to give you `3.0`. When you convert a float to an int, you remove the decimal point and everything after it. To get the whole number part of a division, you can convert the float result to an int.

Quick check 6.3

Write a line of code for each of the following points and then answer the questions at the end:

1

Initialize a variable named `stars` with the value `50`.

2

Initialize another variable named `stripes` with the value `13`.

3

Initialize another variable named `ratio` with the value `stars` divided by `stripes`. Question: what is the type of `ratio`?

4

Convert `ratio` to an int and save the result in a variable named `ratio_truncated`. Question: what is the type of `ratio_truncated`?

In the given task, you'll divide the minutes by 60 and convert the result to an integer to give you the number of whole hours, like so:

```
minutes_to_convert = 123
hours_decimal = minutes_to_convert/60
hours_part = int(hours_decimal)
```

At this point, your `hours_part` variable holds the number of hours converted from the input.

6.3.2. How many minutes?

Finding out the number of minutes is a little bit trickier. In this section, you'll see two ways of doing this:

- *Method 1*—Use the decimal portion of the result from the division. If you use the 123 minutes example, how can you convert the decimal part 0.05 into minutes? You should multiply 0.05 by 60 to give you 3.
- *Method 2*—Use the remainder operator, `%`. Again, use the 123 minutes example. The remainder when 123 is divided by 60 is 3.

6.4. YOUR FIRST PYTHON PROGRAM: ONE SOLUTION

The code for the final program using method 1 is shown in listing 6.1. The code is separated into four parts. The first part initializes the variable to hold the given number of minutes to convert. The second converts the given input into a whole number of hours. The third converts the given input into the whole number of minutes. The last part prints the results.

Listing 6.1. Convert minutes to hours and minutes using the decimal part

```
minutes_to_convert = 123
hours_decimal = minutes_to_convert/60              1
hours_part = int(hours_decimal)                    1

minutes_decimal = hours_decimal-hours_part         2
minutes_part = round(minutes_decimal*60)           2

print("Hours")                                     3
print(hours_part)                                  3
print("Minutes")                                   3
```

- ***1*** **Finds the decimal version of the number of hours and gets the whole number of hours by converting to an int type**
- ***2*** **Gets the part after the decimal point and converts it to whole minutes**
- ***3*** **Prints the results**

The part where you calculate the number of minutes from the decimal number may look a bit intimidating, but you can break it down to understand what's going on. The following line gets the part after the decimal point from the division:

```
minutes_decimal = hours_decimal-hours_part
```

For the example, if `minutes_to_convert` is 123, this calculates as `minutes_decimal = hours_decimal -hours_part = 2.05 – 2 = 0.05`. You now have to convert the 0.05 into minutes.

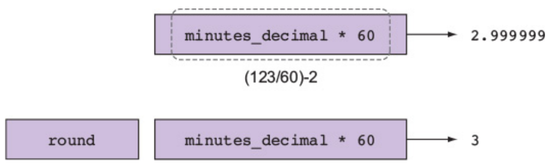
The following line is made up of two separate operations, as shown in figure 6.2:

```
minutes_part = round(minutes_decimal * 60)
```

First it multiplies `minutes_decimal * 60`:

Then it rounds that result with `round(minutes_decimal * 60)`.

Figure 6.2. The two calculations and their evaluations, in order, on the variable `minutes_decimal`



Why do you need to do all these operations? If you run the program with the line

```
minutes_part = minutes_decimal * 60
```

instead of

```
minutes_part = round(minutes_decimal * 60)
```

you'll notice something interesting. The output is

```
Hours
2
Minutes
2.9999999999999893
```

You expected to see 3 but see 2.999999999893. What's going on? This behavior occurs because of the way that Python stores floats. Computers can't store decimal numbers precisely because they can't represent fractions exactly. When they represent 0.05 in memory, they approximate this number. When you multiply floats, the small differences between their exact value and how they're represented in memory are amplified.

When you multiply 0.05 by 60, the result is off by 0.0000000000000107. You can solve this by rounding your final answer to an integer with `round(minutes_decimal * 60)`.

Quick check 6.4

Q1:

Change the program in listing 6.1 to find the hours and minutes when starting with 789 minutes. What's the output?

6.5. YOUR FIRST PYTHON PROGRAM: ANOTHER SOLUTION

The code for the final program using method 2 is shown in the next listing. The code is separated into the same four parts as the previous solution: initializing, getting the whole number of hours, getting the whole number of minutes, and printing the result.

Listing 6.2. Convert minutes to hours and minutes using the remainder

```
minutes_to_convert = 123

hours_decimal = minutes_to_convert/60
hours_part = int(hours_decimal)

minutes_part = minutes_to_convert%60

print("Hours")
print(hours_part)
print("Minutes")
print(minutes_part)
```

- ***1*** **The given number of minutes**
- ***2*** **Finds the decimal version of the number of hours**
- ***3*** **Gets the whole number of hours by converting to an int type**
- ***4*** **Uses the remainder when you divide the number of minutes by 60 to get the whole minutes**

The output of this program is as follows:

```
Hours
2
Minutes
3
```

This version of the program uses the remainder idea to give a more concise program in which you don't need to do any "post-processing" to round or convert to integers, as you had to do with the previous method. But good style would be to leave a comment right above the line `minutes_part = minutes_to_convert % 60` to remind yourself that the remainder when divided by 60 gives you the whole number of minutes. An appropriate comment is shown here:

```
# the remainder gives the number of minutes remaining
```

SUMMARY

In this lesson, my objective was to teach you how to put together many ideas to write your first Python program. The program incorporated the following main ideas:

- Thinking about the given task and dividing it into a few smaller tasks
- Creating variables and initializing them to a value
- Performing operations on variables
- Converting variable types to other types
- Printing output to the user

Let's see if you got this ...

Q6.1

Write a program that initializes a variable with the value 75 to represent the temperature in Fahrenheit. Then convert that value into Celsius by using the formula c = (f - 32) / 1.8. Print the Celsius value.

Q6.2

Write a program that initializes a variable with the value 5 to represent a number of miles. Then convert this value into kilometers and then meters by using km = miles / 0.62137 and meters = 1000 * km. Print the result in the following form:

```
miles
5
km
8.04672
meters
8046.72
```