Python Essentials 1:
Module 2
Data types, variables, basic input-output operations, basic operators

In this module, you will learn:

how to write and run simple Python programs;
what Python literals, operators, and expressions are;
what variables are and what are the rules that govern them;
how to perform basic input and output operations.

(part 5)

Leaving comments in code: why, how, and when

You may want to put in a few words addressed not to Python but to humans, usually to explain to other readers of the code how the tricks used in the code work, or the meanings of the variables, and eventually, in order to keep stored information on who the author is and when the program was written.

A remark inserted into the program, which is omitted at runtime, is called a comment.

How do you leave this kind of comment in the source code? It has to be done in a way that won't force Python to interpret it as part of the code.

Whenever Python encounters a comment in your program, the comment is completely transparent to it - from Python's point of view, this is only one space (regardless of how long the real comment is).

In Python, a comment is a piece of text that begins with a # (hash) sign and extends to the end of the line.

If you want a comment that spans several lines, you have to put a hash in front of them all.

Just like here:

```
# This program evaluates the hypotenuse c.
# a and b are the lengths of the legs.
a = 3.0
b = 4.0
c = (a ** 2 + b ** 2) ** 0.5  # We use ** instead of square root.
print("c =", c)
```

Good, responsible developers describe each important piece of code, e.g., explaining the role of the variables; although it must be stated that the best way of commenting variables is to name them in an unambiguous manner.

For example, if a particular variable is designed to store an area of some unique square, the name square_area will obviously be better than aunt_jane.

We say that the first name is self-commenting.

Comments may be useful in another respect - you can use them to mark a piece of code that currently isn't needed for whatever reason. Look at the example below, if you uncomment the highlighted line, this will affect the output of the code:

```
# This is a test program.
x = 1
y = 2
# y = y + x
print(x + y)
```

This is often done during the testing of a program, in order to isolate the place where an error might be hidden.

TIP

If you'd like to quickly comment or uncomment multiple lines of code, select the line(s) you wish to modify and use the following keyboard shortcut: CTRL + / (Windows) or CMD + / (Mac OS). It's a very useful trick, isn't it? Try this code in Sandbox.

LAB

Estimated time
5 minutes

Level of difficulty
Very Easy

Objectives
becoming familiar with the concept of comments in Python;
using and not using comments;
replacing comments with code;
experimenting with Python code.
Scenario
The code in the editor contains comments. Try to improve it: add or remove comments where you find it appropriate (yes, sometimes removing a comment can make the code more readable), and change variable names where you think this will improve code comprehension.

NOTE

Comments are very important. They are used not only to make your programs easier to understand, but also to disable those pieces of code that are currently not needed (e.g., when you need to test some parts of your code only, and ignore other). Good programmers describe each important piece of code, and give self-commenting names to variables, as sometimes it is simply much better to leave information in the code.

It's good to use readable variable names, and sometimes it's better to divide your code into named pieces (e.g., functions). In some situations, it's a good idea to write the steps of computations in a clearer way.

One more thing: it may happen that a comment contains a wrong or incorrect piece of information - you should never do that on purpose!

Key takeaways

1. Comments can be used to leave additional information in code. They are omitted at runtime. The information left in source code is addressed to human readers. In Python, a comment is a piece of text that begins with #. The comment extends to the end of line.

2. If you want to place a comment that spans several lines, you need to place # in front of them all. Moreover, you can use a comment to mark a piece of code that is not needed at the moment (see the last line of the snippet below), e.g.:

```
# This program prints
# an introduction to the screen.
print("Hello!")  # Invoking the print() function
# print("I'm Python.")
```

3. Whenever possible and justified, you should give self-commenting names to variables, e.g., if you're using two variables to store a length and width of something, the variable names length and width may be a better choice than myvar1 and myvar2.

4. It's important to use comments to make programs easier to understand, and to use readable and meaningful variable names in code. However, it's equally important not to use variable names that are confusing, or leave comments that contain wrong or incorrect information!

5. Comments can be important when you are reading your own code after some time (trust us, developers do forget what their own code does), and when others are reading your code (can help them understand what your programs do and how they do it more quickly).

Exercise 1

What is the output of the following snippet?

```
# print("String #1")
print("String #2")
```

Exercise 2

What will happen when you run the following code?

```
# This is
a multiline
comment. #

print("Hello!")
```