

# Introduction to Python

In this course, you will work in teams of 3–4 students to learn new concepts. This activity will introduce you to the process. We'll take a first look at variables, assignment, and input/output.

## Content Learning Objectives

*After completing this activity, students should be able to:*

- Describe differences between program and output text.
- Identify and execute Python functions for input/output.
- Write assignment statements and use assigned variables.

## Process Skill Goals

*During the activity, students should make progress toward:*

- Leveraging prior knowledge and experience of other students. (Teamwork)

## Facilitation Notes

During the first 5–10 minutes, help teams get started by encouraging discussion and collaboration. Make sure that each team is able to find Thonny (or other development environment) on their computer. If multiple interpreters are installed (e.g., Python 2 and Python 3), double check that students run the correct version.

Keep a close watch on the first few questions, and encourage teams that make mistakes to go back and rework them (without giving the answer). Once most teams have successfully run the program in **Model 1**, type or open the same code on the instructor's machine. Report out questions **#4** and **#7**, using the projected program as a visual aid.

On **Model 2**, make sure that teams use the Shell window (some teams might accidentally use the Editor window). Once most teams have started working on the table, you might want to demonstrate on the projector how the Shell works. Report out the last three questions, and discuss how values from user input are different from literal values in the source code.

Before students begin **Model 3**, demonstrate how to read error messages. They do not need to write down the entire error message, but they should identify the type of error (e.g., `NameError`, `SyntaxError`). It may help to make several mistakes on the projector and show students how to find the first word at the end of the error message.

At the end of the activity, review how variables work in the Shell. Demonstrate and explain the following lines: `x = 10`, `x (by itself)`, `x + 1`, `x (by itself)`, `x = x + 1`, and `x (by itself)`.

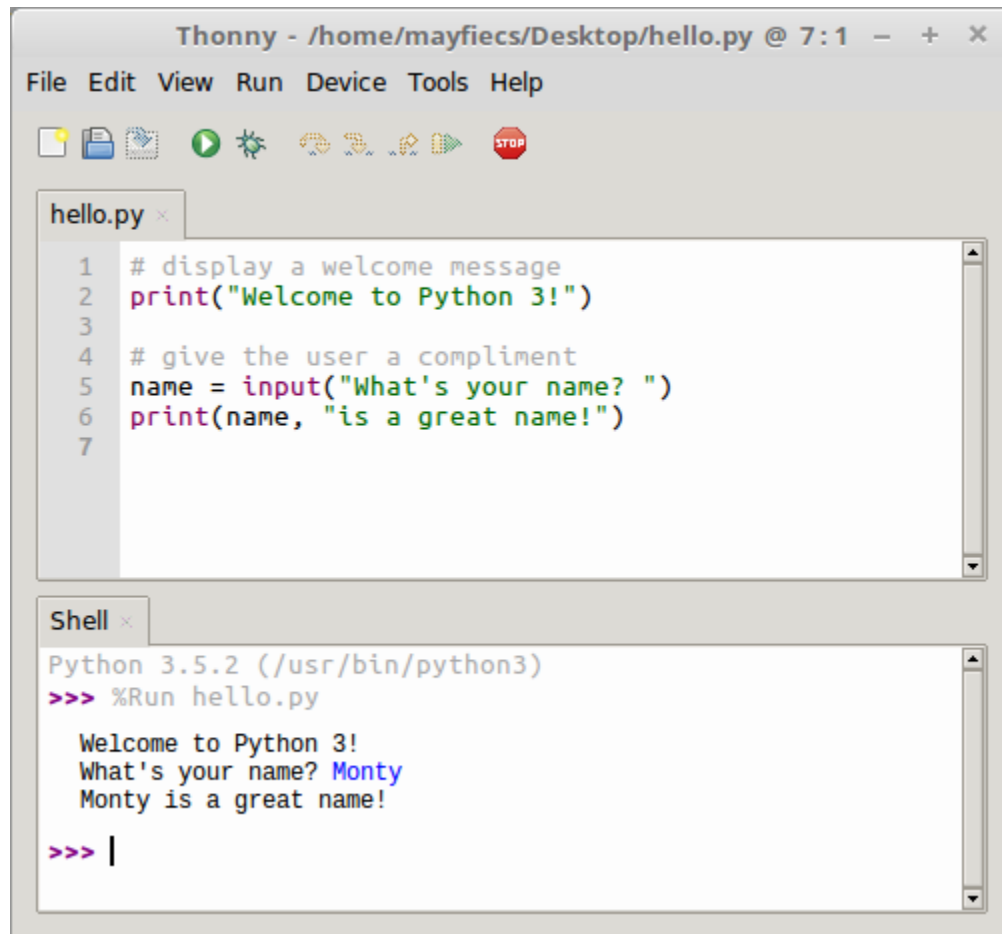


Copyright © 2019 C. Mayfield, T. Shepherd, and H. Hu. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

# Model 1 Getting Started with Thonny

Thonny is an integrated development environment (IDE) for Python designed for learning and teaching programming. It is freely available at <https://thonny.org/>.

**Do not run Thonny yet! Answer the questions first!**



Questions (15 min)

Start time: \_\_\_\_\_

1. Based on the screenshot in Model 1:

a) where is the *Shell* window? the second / bottom half

b) where is the *Editor* window? the first / top half

c) what is the name of the file in the Editor? hello.py

d) what is the directory where this file is located? Desktop

2. Identify the number of lines corresponding to:

a) the program (in the Editor)

The source file contains six total lines; the 7th line is blank and doesn't count.

b) the output of the program

The program displays three lines ("Welcome...", "What's...", and "Monty...").

3. What is the symbol at the start of a line of program text not displayed as output?

The pound sign: #

4. Consider the three program lines (in the Editor) that are not displayed as output. Describe what might be the purpose of:

a) a **comment** line (starts with a pound sign: #)

They are used to describe the purpose and/or behavior of the code.

b) a blank line

They make the code easier to read.

*Now open Thonny on your computer, type the code shown in Model 1, save the file as hello.py, and run the program. Ask for help if you get stuck!*

5. What was required before the third line of the program output was displayed?

An answer needed to be entered (typed).

6. In the Shell window, what is the color of:

a) the program's output? black

b) the user's input? blue

7. Based on your experience so far, what is the difference between the text in the Editor window and the text in the Shell window?

The editor contains a series of statements or commands which make up the program. The shell (or terminal) contains the program output text and an interface for program input.

8. Describe what appears to be the purpose of each line of Python code in the Editor window.

a) line 1: a comment to describe the purpose of the code

b) line 2: displays a message

c) line 3: a blank line to make the code easier to read

d) line 4: a comment to describe the purpose of the code

e) line 5: gets input from the user

f) line 6: displays the final output

## Model 2 Python Built-In Functions

You can use *functions* to perform specific operations. Some functions require values, known as *arguments*, to perform their operation. Functions may also *return* a result. For example:

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

`input` is a function, `"What's your name? "` is an argument, and the return value (typed by the user) is stored in `name`.

The following table shows additional examples of functions. They were written by a scientist to set up an experiment.

**Do not type anything yet! Read the questions first!**

Python code	Shell output
<code>input("enter the mass in grams: ")</code>	enter the mass in grams: 100
<code>mass = input("enter another mass in grams: ")</code>	enter another mass in grams: 10
<code>mass</code>	'10'
<code>unit = input("enter the units for mass: ")</code>	enter the units for mass: g
<code>print(mass, unit)</code>	10 g
<code>print(mass / 2)</code>	TypeError
<code>ten = 10</code>	
<code>print(ten / 2)</code>	5.0
<code>abs(-1)</code>	1
<code>abs(-1 * ten)</code>	10

## Questions (15 min)

Start time: \_\_\_\_\_

9. List the names of the three functions used in Model 2.

input, print, abs

10. What are the arguments of the first use of the `print` function? `mass, unit`

11. Type each line of code in a Python Shell, one line at a time, and write the corresponding output (if observed) in the right column of the table. If an error occurs, write what type of error it was (i.e., the first word of the last line of the error message).

Place an asterisk (\*) next to any output for which you were surprised, and note what was unexpected about the output. Don't worry yet about *understanding* any strange output you may see; we will discuss what it all means by the end of class.

12. Which function delayed execution until additional input was entered?

The input function.

13. Which term, *user* or *programmer*, best defines the role of the person who entered the additional input? Explain.

User is a better term; programs are ultimately written to be used by non-programmers.

14. Based on the Shell output, what does the word `mass` represent, and how did it get its value?

(Answers may vary) Its value is 10, which the user entered as input.

15. What does the word `ten` represent, and how did it get its value?

Its value is 10 based on the statement "`ten = 10`".

16. Do the values of `mass` and `ten` both represent a number? Explain why or why not.

Although they both appear to represent a numerical value, `mass` divided by 2 gives an error, while `five` divided by 2 gives the expected numerical outcome.

## Model 3 Variables and Assignment

In programming, an *assignment statement* saves a value to a *variable*. The variable “is set to” the value after the “=” *operator*. For example:

```
mass = input("enter the mass in grams: ")
```

Selecting concise yet descriptive variable names is considered good programming style and will make your programs easier to read. Consider the examples in the table below.

**Do not type anything yet! Read the questions first!**

Python code	Shell output
data = 12	
data	12
Data	NameError
Data = 34	
data	12
Data	34
my data = 56	SyntaxError
my_data = 78	
3data = "hello"	SyntaxError
data3 = "world"	
data3 = hello	NameError
mass = 273 + 100	
273 + 100 = mass	SyntaxError
mass	373
Mass + 100	NameError
mass - 100	273

### Questions (15 min)

Start time: \_\_\_\_\_

17. Pick one assignment statement from the table above, and identify the following:

a) the variable being assigned

b) the assignment operator

c) the value of the variable immediately after the assignment

18. Similar to Model 2, type each line of code in a Python Shell and write the corresponding output in the space above. If an error occurs, write what type of error. Place an asterisk (\*) next to any output for which you were surprised.

19. Circle each *successful* assignment statement in Model 3. How many are there? 5

20. What is the observed output of a successful assignment statement?

There is no error message (there is no output at all).

21. After the successful execution of an assignment statement, how can you confirm the value of this variable?

You can type back the variable name to see its current value.

22. For each assignment statement that executed without an error, write the corresponding variable name.

data, Data, my\_data, data3, mass  
(see the lines with no output)

23. Based on the Model 3 output, indicate whether each statement below is true or false.

a) Variable names in Python can start with a number. false

b) Variable names in Python must start with a lower-case letter. false

c) Variable names in Python may not include spaces. true

d) Variable names in Python are case-sensitive. true

24. Each of the following assignment statements has an error. Write a valid line of Python code that corrects the assignment statement. Double-check your code using a computer.

a) `3 + 4 = answer` `answer = 3 + 4`

b) `oh well = 3 + 4` `oh_well = 3 + 4`

c) `2x = 7` `x2 = 7`

**25.** Predict the value of the variable `mass` after executing all lines of code in Model 3. Then test your prediction on a computer, and explain the result.

The value is 373, because `mass` was only assigned one time. The other lines did not change the value of `mass`.

**26.** Write a line of Python code to assign the current value of `mass` to the variable `temp`. Show output that confirms that you have done this correctly, and explain the code.

The correct line is `temp = mass`. To verify the result, simply type `temp`. The assignment operation reads from right to left.