

Fundamentals of Python Programming

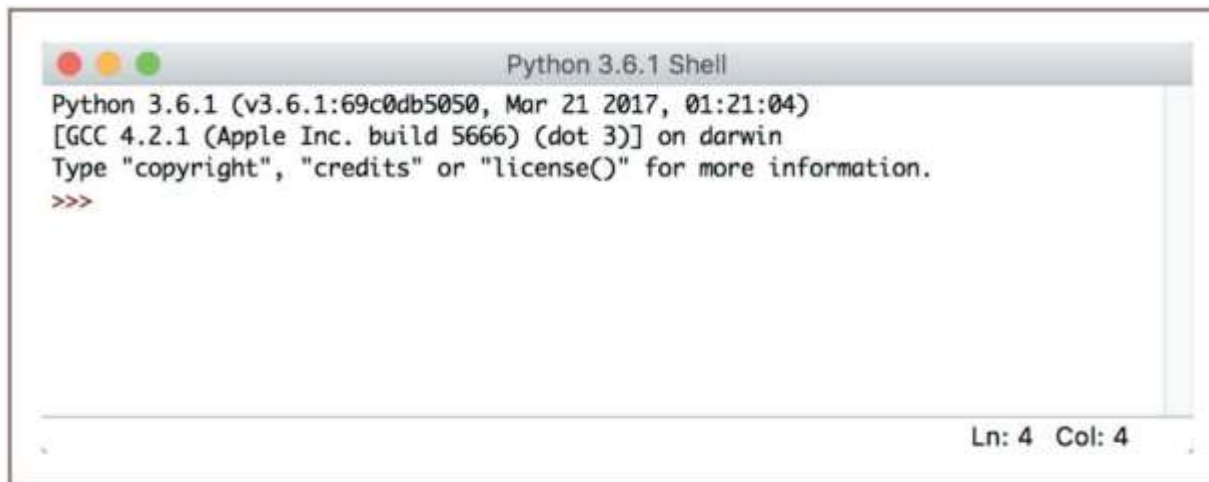
Getting Started with Python Programming

- Early 1990s: Guido van Rossum
 - invented the Python programming language
- **Python** is a high-level, general-purpose programming language for solving problems on modern computer systems
- Useful resources at www.python.org

Running Code in the Interactive Shell (1 of 2)

- Python is an **interpreted** language
- Simple Python expressions and statements can be run in the **shell**
 - Easiest way to open a Python shell is to launch the IDLE
 - To quit, select the window's close box or press Control+D
 - Shell is useful for:
 - Experimenting with short expressions or statements
 - Consulting the documentation

Running Code in the Interactive Shell (2 of 2)

A screenshot of a macOS-style window titled "Python 3.6.1 Shell". The window has a title bar with three colored buttons (red, yellow, green) on the left. The main content area displays the following text: "Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)", "[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin", "Type \"copyright\", \"credits\" or \"license()\" for more information.", and a red prompt ">>>". The status bar at the bottom right shows "Ln: 4 Col: 4".

```
Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

Ln: 4 Col: 4

Figure 1-6 Python shell window



Input, Processing, and Output (1 of 5)

- Programs usually accept inputs from a source, process them, and output results to a destination
 - In terminal-based interactive programs, these are the keyboard and terminal display
- In Python, inputs are Python expressions or statements
 - Outputs are the results displayed in the shell
- Programmers can also force output of a value by using the print function
 - **print (<expression>)**
- Example:

```
>>>print ("Hi there")  
Hi there
```



Input, Processing, and Output (2 of 5)

- The following example receives an input string from the user and saves it for further processing:

```
>>> name = input("Enter your name:")  
Enter your name: Ken Lambert  
>>> name  
'Ken Lambert'  
>>> print(name)  
Ken Lambert  
>>>
```

Input, Processing, and Output (3 of 5)

- The **input** function always builds a string from the user's keystrokes and returns it to the program
- Strings that represent numbers must be converted from strings to appropriate number types
 - Two type conversion functions: **int** (for integers) and **float** (for floating-point numbers)



Input, Processing, and Output (4 of 5)

- The next session inputs two integers and displays their sum:

```
>>> first = int(input("Enter the first number: "))  
Enter the first number: 23  
>>> second = int(input("Enter the second number:"))  
Enter the second number: 44  
>>> print("The sum is", first + second)  
The sum is 67
```

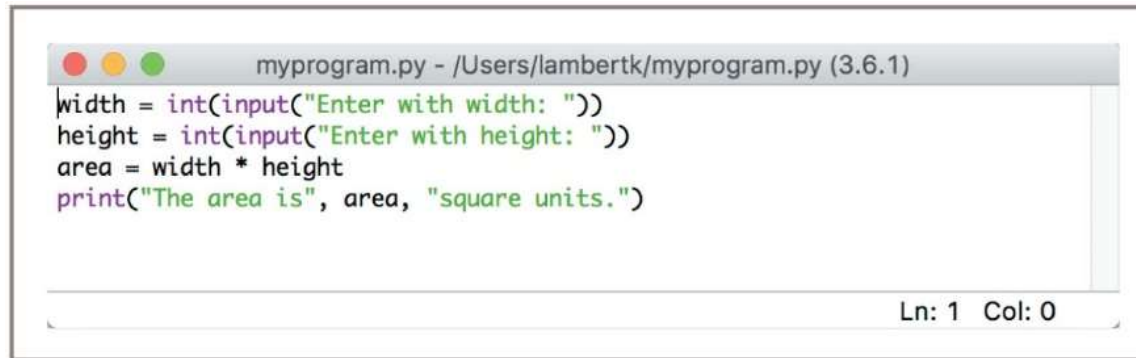

Input, Processing, and Output (5 of 5)

Function	What It Does
<code>float(<a <i>string of digits</i>>)</code>	Converts a string of digits to a floating-point value.
<code>int(<a string of digits>)</code>	Converts a string of digits to an integer value.
<code>input(<a <i>string prompt</i>>)</code>	Displays the string prompt and waits for keyboard input. Returns the string of characters entered by the user.
<code>print(<expression>, ..., <expression>)</code>	Evaluates the expressions and displays them, separated by one space, in the console window.
<code><string 1> + <string 2></code>	Glues the two strings together and returns the result.

Editing, Saving, and Running a Script (1 of 3)

- We can then run Python program files or **scripts** within IDLE or from the OS's command prompt
 - Run within IDLE using menu option, F5 (Windows), or Control+F5 (Mac or Linux)
- Python program files use **.py** extension
- Running a script from IDLE allows you to construct some complex programs, test them, and save them in **program libraries** to reuse or share with others

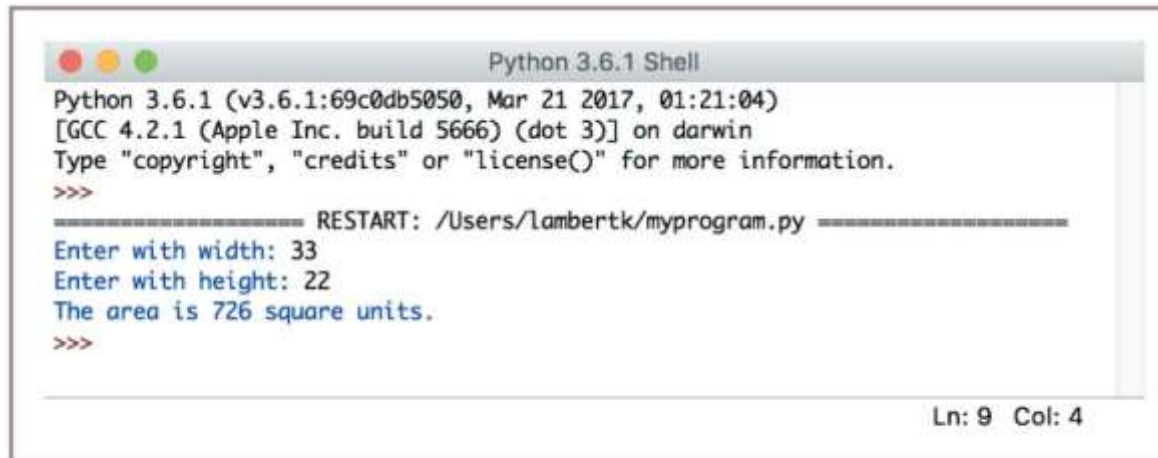
Editing, Saving, and Running a Script (2 of 3)

A screenshot of a Python script editor window. The window has a title bar with three colored buttons (red, yellow, green) and the text "myprogram.py - /Users/lambertk/myprogram.py (3.6.1)". The main area contains four lines of Python code: "width = int(input('Enter with width: '))", "height = int(input('Enter with height: '))", "area = width * height", and "print('The area is', area, 'square units.')." The code is color-coded: keywords are purple, strings are green, and operators/variables are black. A status bar at the bottom right shows "Ln: 1 Col: 0".

```
myprogram.py - /Users/lambertk/myprogram.py (3.6.1)
width = int(input("Enter with width: "))
height = int(input("Enter with height: "))
area = width * height
print("The area is", area, "square units.")
Ln: 1 Col: 0
```

Figure 1-7 Python script in an IDLE window

Editing, Saving, and Running a Script (3 of 3)

A screenshot of a macOS-style window titled "Python 3.6.1 Shell". The window contains the following text: "Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)", "[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin", "Type 'copyright', 'credits' or 'license()' for more information.", and a prompt ">>>". Below this, a separator line is followed by "RESTART: /Users/lambertk/myprogram.py". Then, the user input "Enter with width: 33" is shown, followed by "Enter with height: 22", and the output "The area is 726 square units.". Another prompt ">>>" is at the bottom. The status bar at the bottom right shows "Ln: 9 Col: 4".

```
Python 3.6.1 Shell
Python 3.6.1 (v3.6.1:69c0db5050, Mar 21 2017, 01:21:04)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/lambertk/myprogram.py =====
Enter with width: 33
Enter with height: 22
The area is 726 square units.
>>>
Ln: 9 Col: 4
```

Figure 1-8 Interaction with a script in a shell window

Behind the Scenes: How Python Works

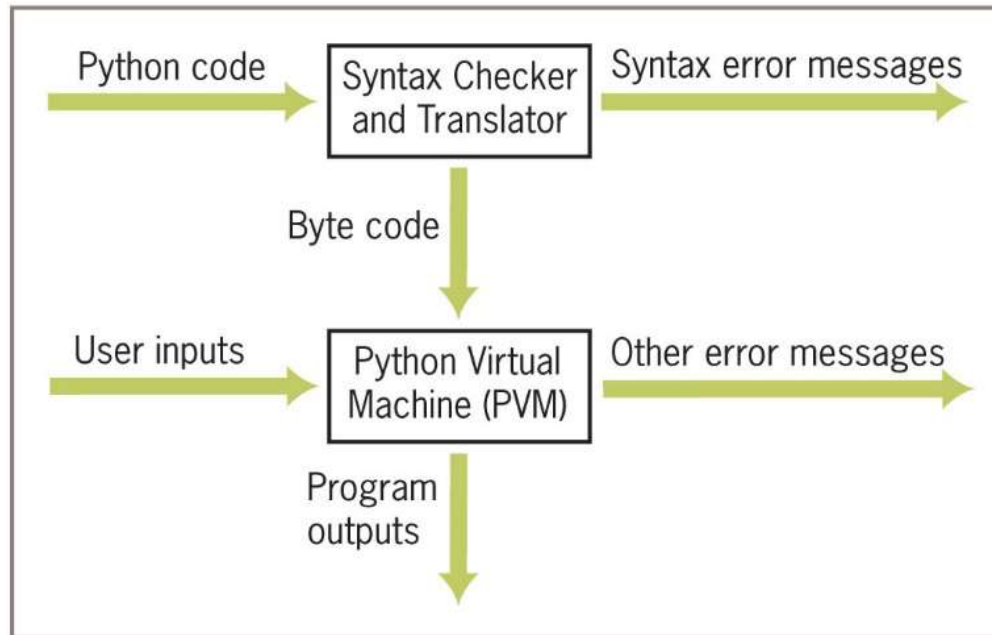


Figure 1-9 Steps in interpreting a Python program



Detecting and Correcting Syntax Errors (1 of 2)

- Programmers inevitably make typographical errors when editing programs, called **syntax errors**
 - The Python interpreter will usually detect these
- **Syntax:** rules for forming sentences in a language
- When Python encounters a syntax error in a program, it halts execution with an error message
- Example:

```
>>> length = int(input("Enter the length: "))
```

```
Enter the length: 44
```

```
>>> print(lenth)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#1>", line 1, in <module>
```

```
NameError: name 'lenth' is not defined
```

Detecting and Correcting Syntax Errors (2 of 2)

- The next statement attempts to print the value of the correctly spelled variable:

```
>>> print(length)
```

SyntaxError: unexpected indent

- Final example, programmer attempts to add two numbers, but forgets to include the second one:

```
>>> 3 +
```

SyntaxError: invalid syntax