TEC222 Introduction to Programming with Python

Lecture 1

Variables and I/O

Starting IDLE

- Windows:
  Invoke with double click of
- MAC:
  Open Finder, select Applications, select the Utilities folder, select Terminal, and then enter IDLE at the prompt
- LINUX and UNIX: usually be found at /usr/bin/idle3

FIGURE 1.15 IDLE tile from Windows.

Starting IDLE

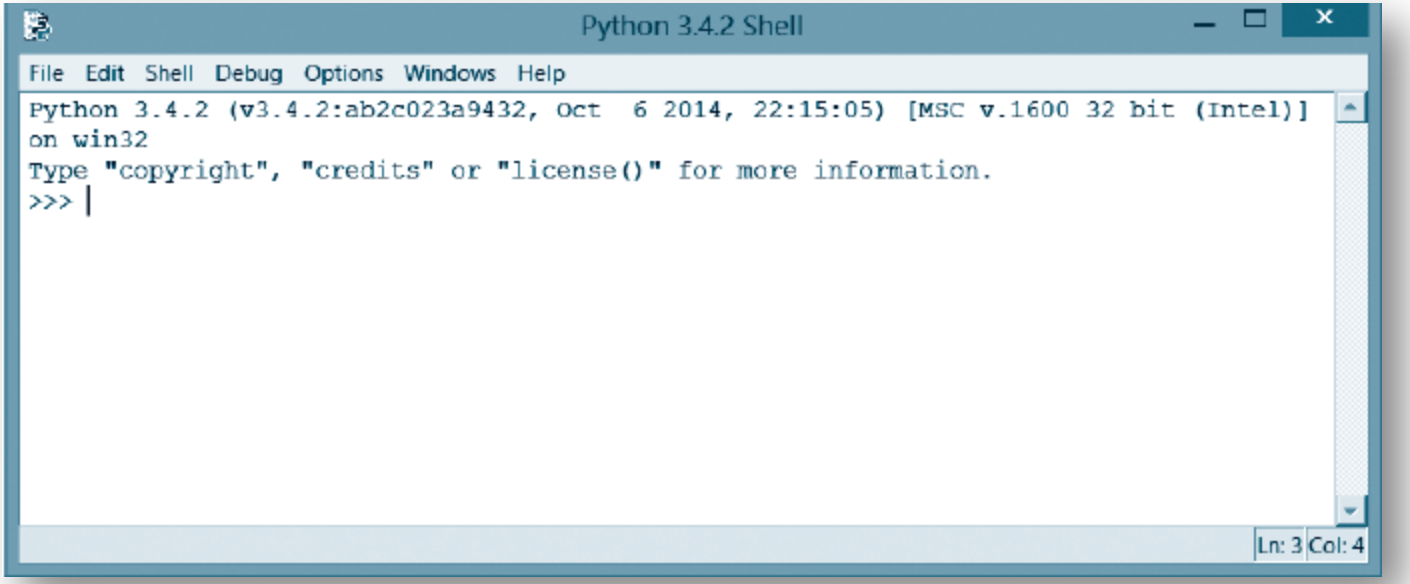


FIGURE 1.16 The Python shell.

Starting IDLE

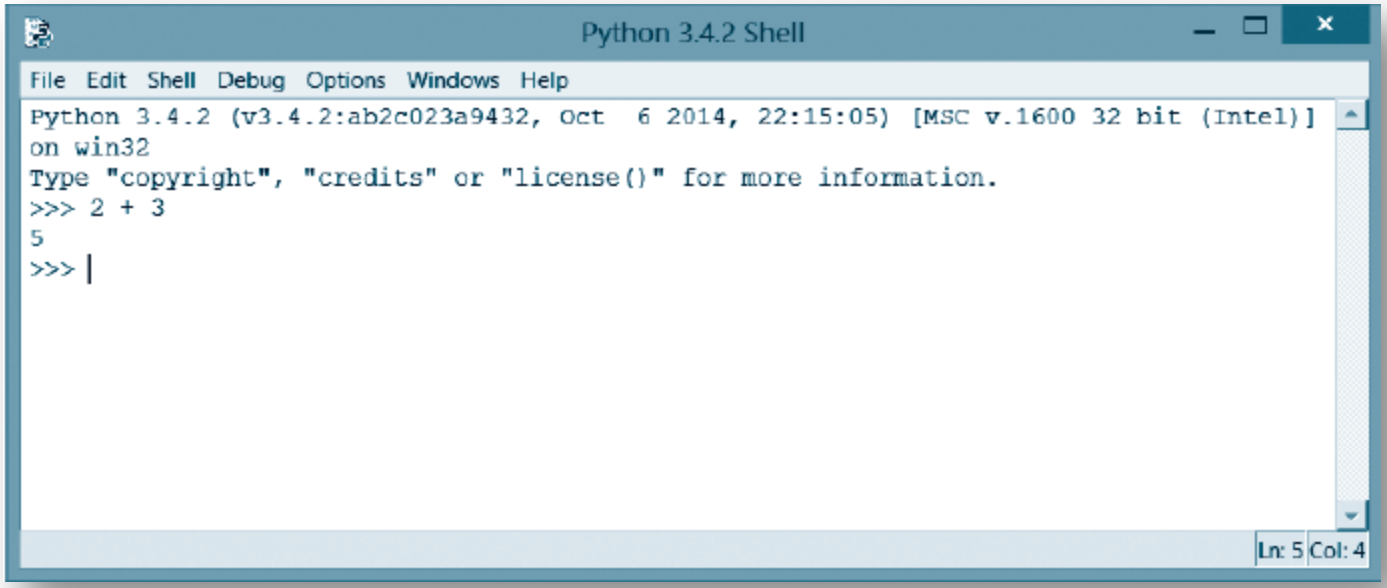


FIGURE 1.17 The Python shell after the expression 2 + 3 has been evaluated.

Starting IDLE

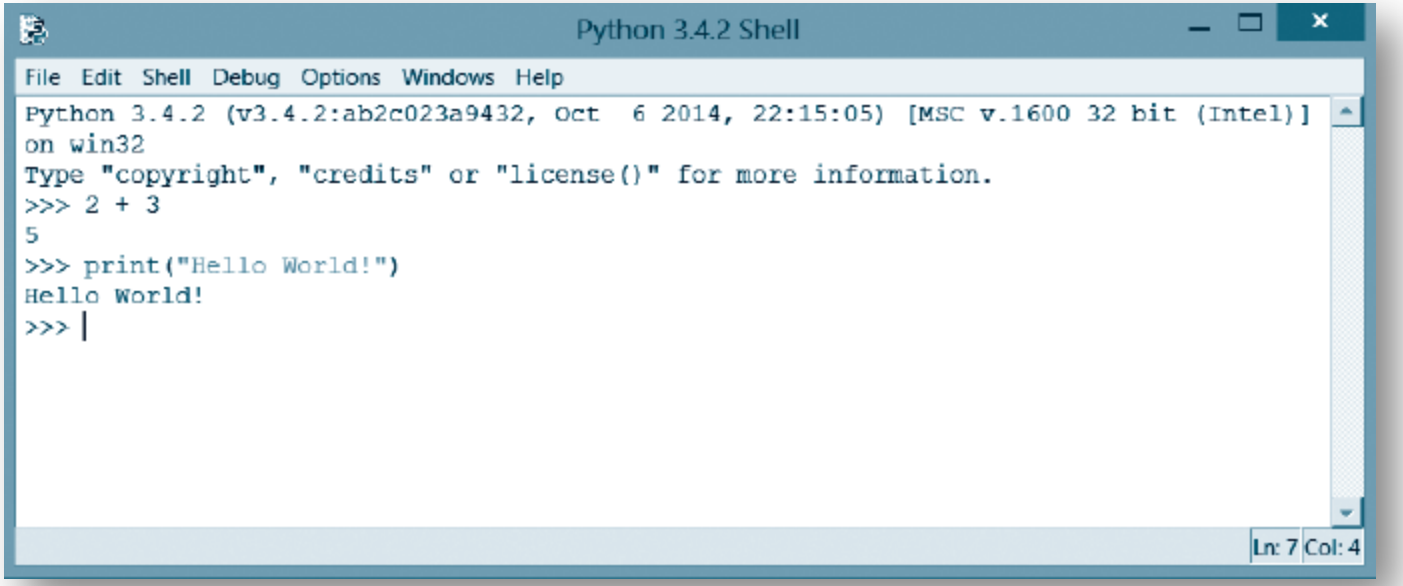


FIGURE 1.18 The Python shell after the statement print("Hello World!") has been executed.

A Python Code Editor Walkthrough

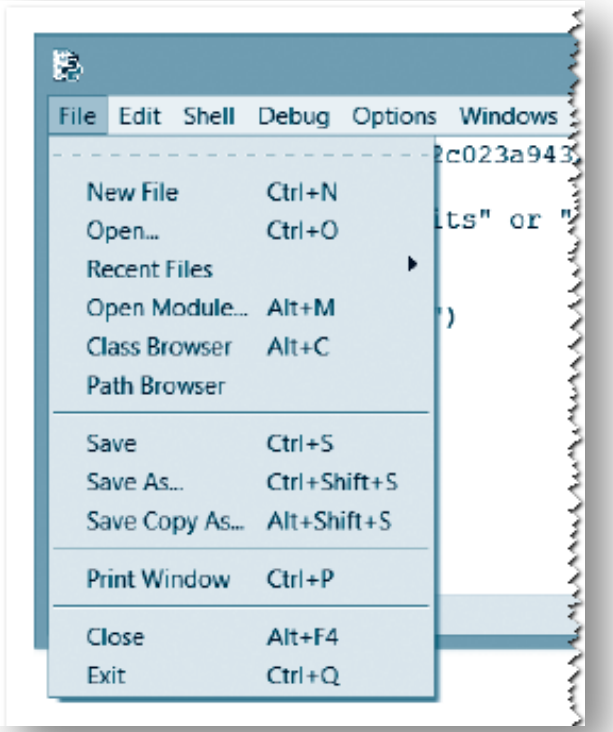


FIGURE 1.19 The File drop-down list.

A Python Code Editor Walkthrough

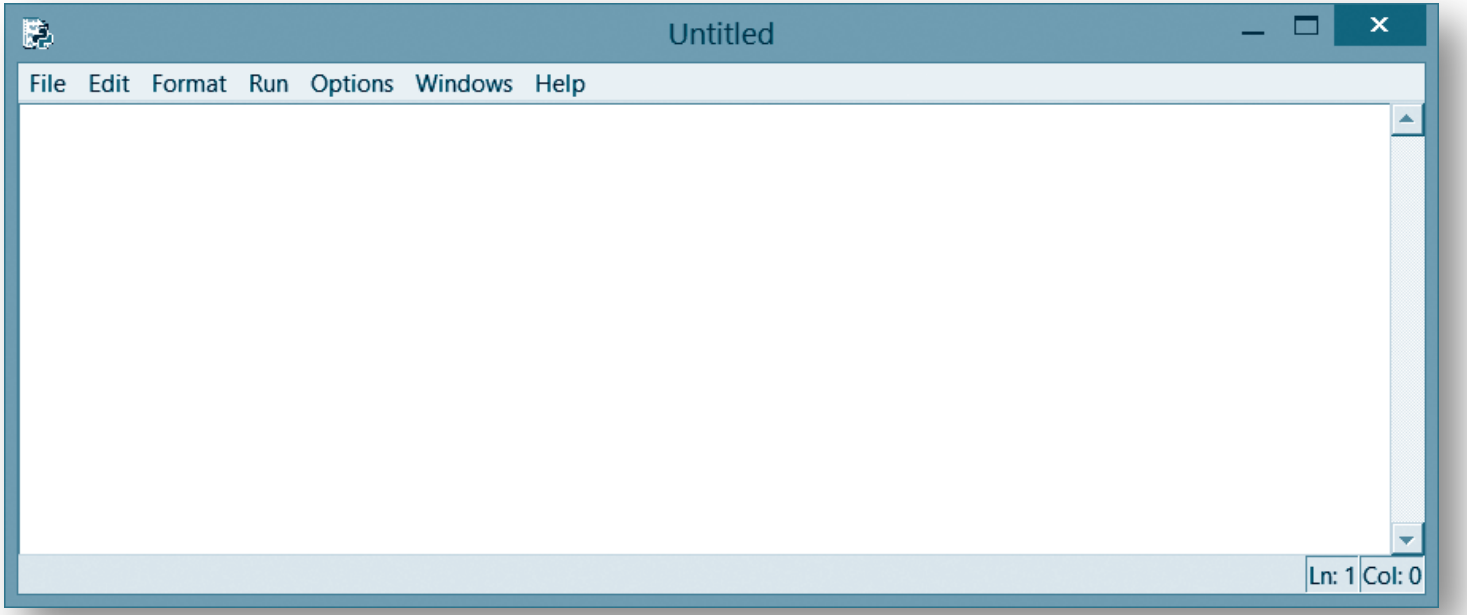


FIGURE 1.20 The code editor window generated after New Window is clicked on.

A Python Code Editor Walkthrough

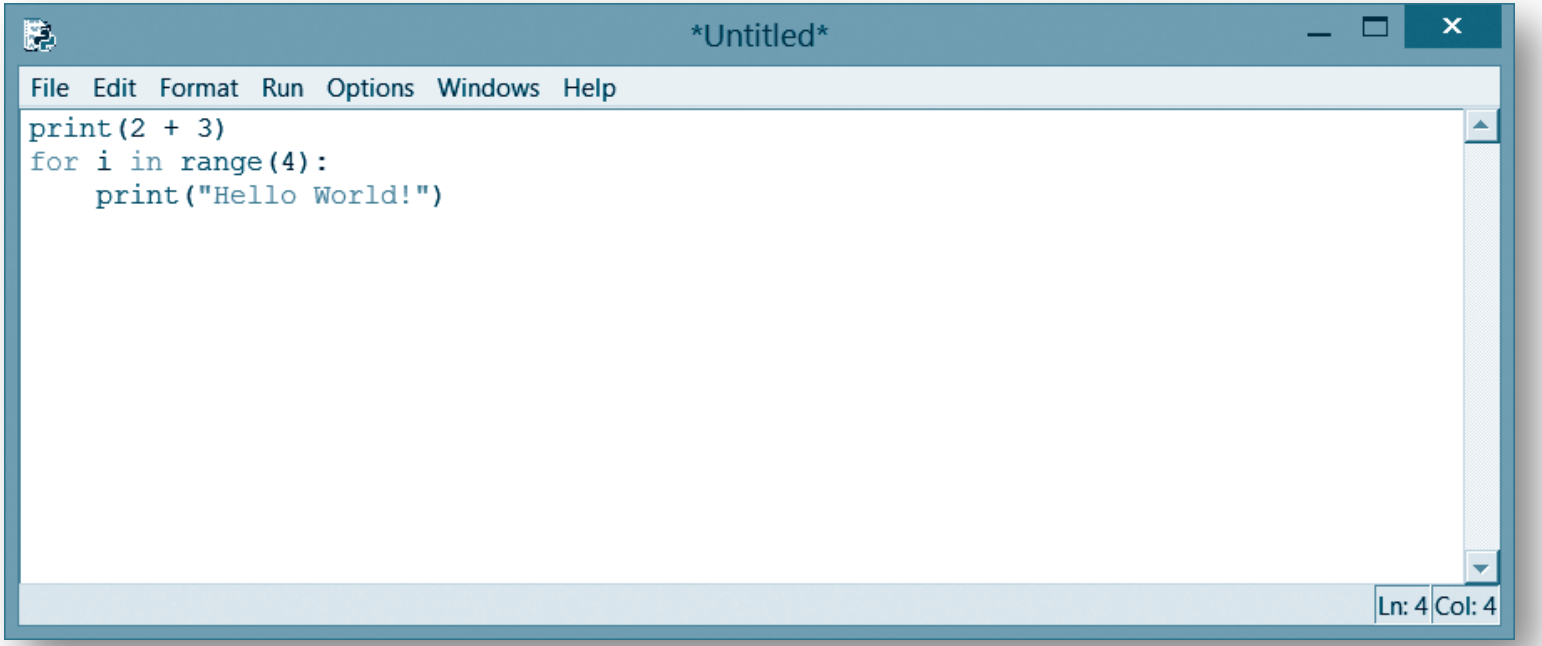


FIGURE 1.21 The code editor window containing a three-line Python program.

A Python Code Editor Walkthrough

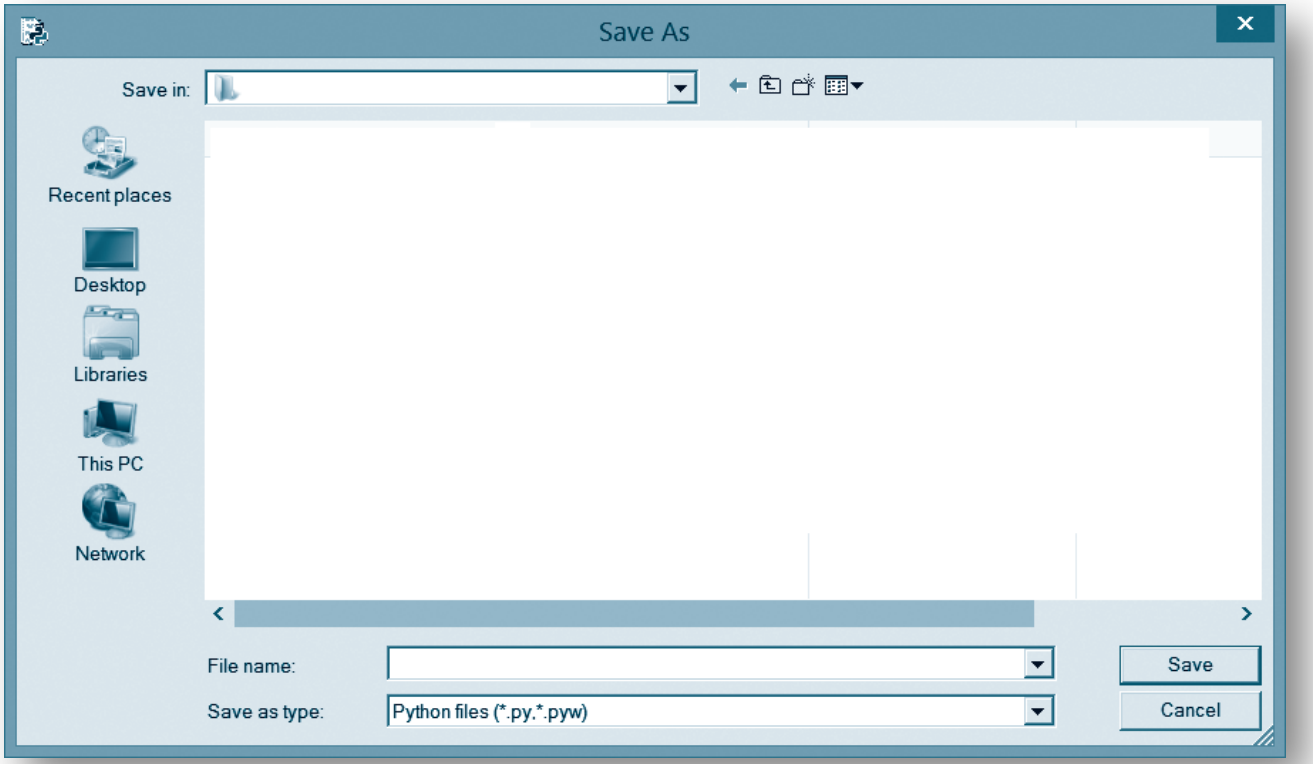


FIGURE 1.22 A Save As dialog box.

A Python Code Editor Walkthrough

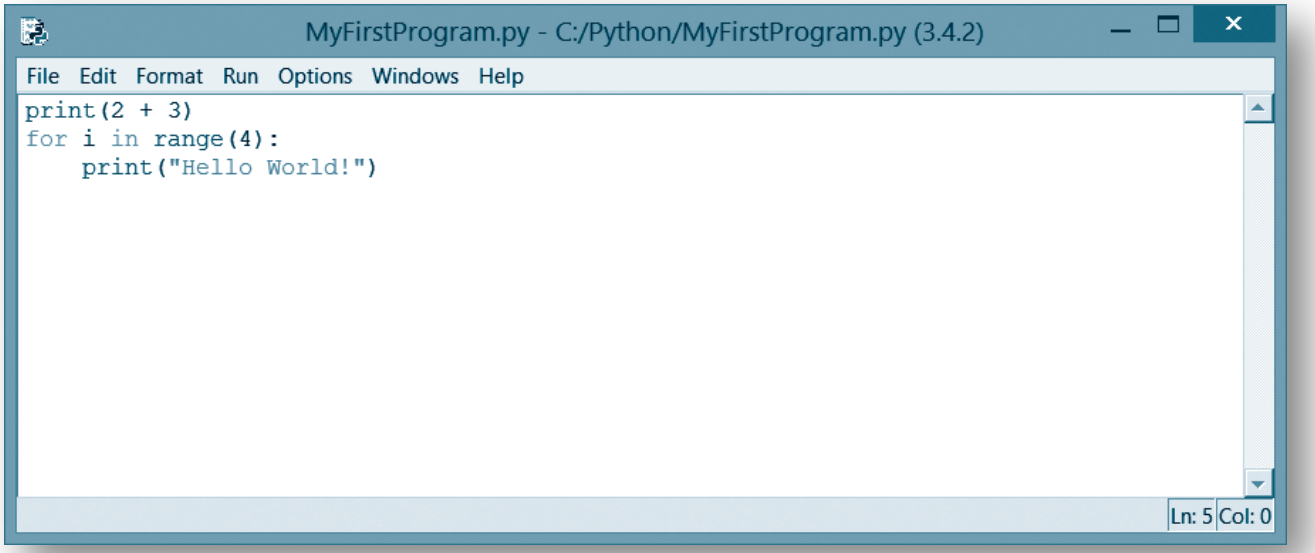


FIGURE 1.23 The code editor window containing a three-line Python program.

A Python Code Editor Walkthrough

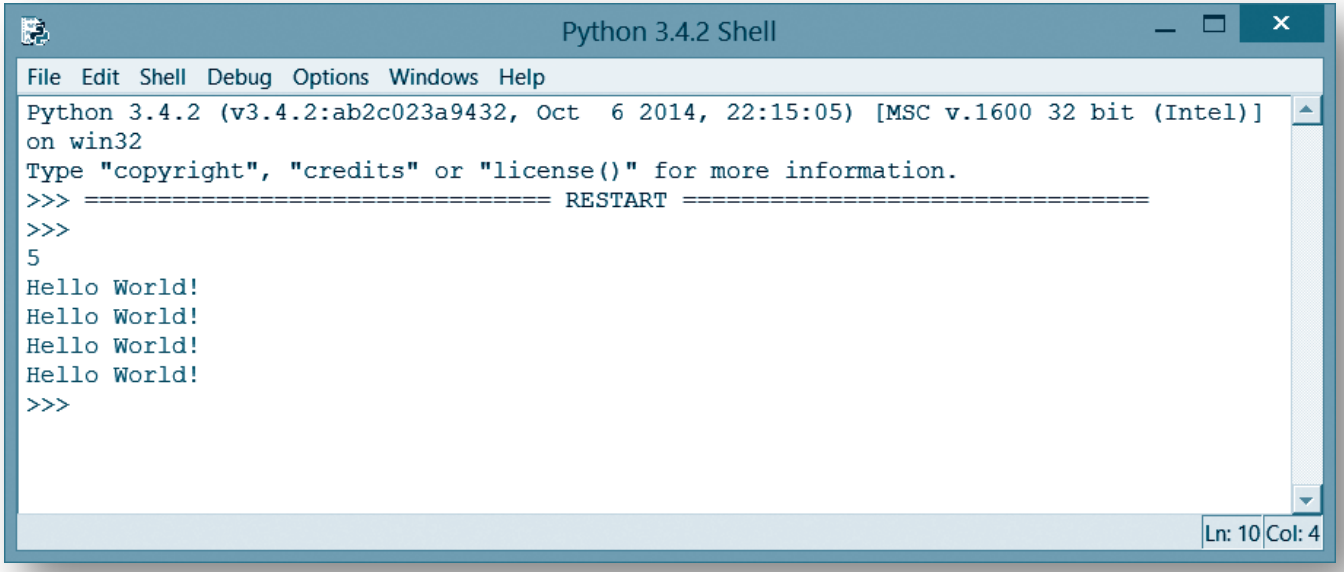


FIGURE 1.24 Press the F5 key to execute.
The outcome of the Python program in Fig. 1.22.

A Python Code Editor Walkthrough

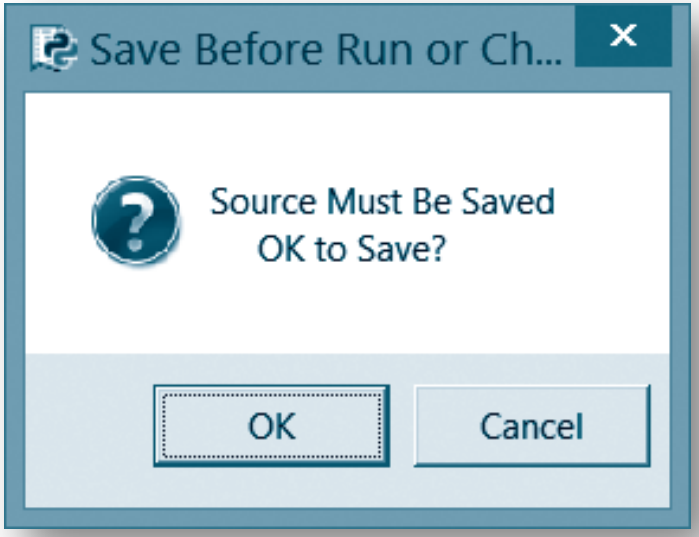


FIGURE 1.25 A Save message box.

An Open-a-Program Walkthrough

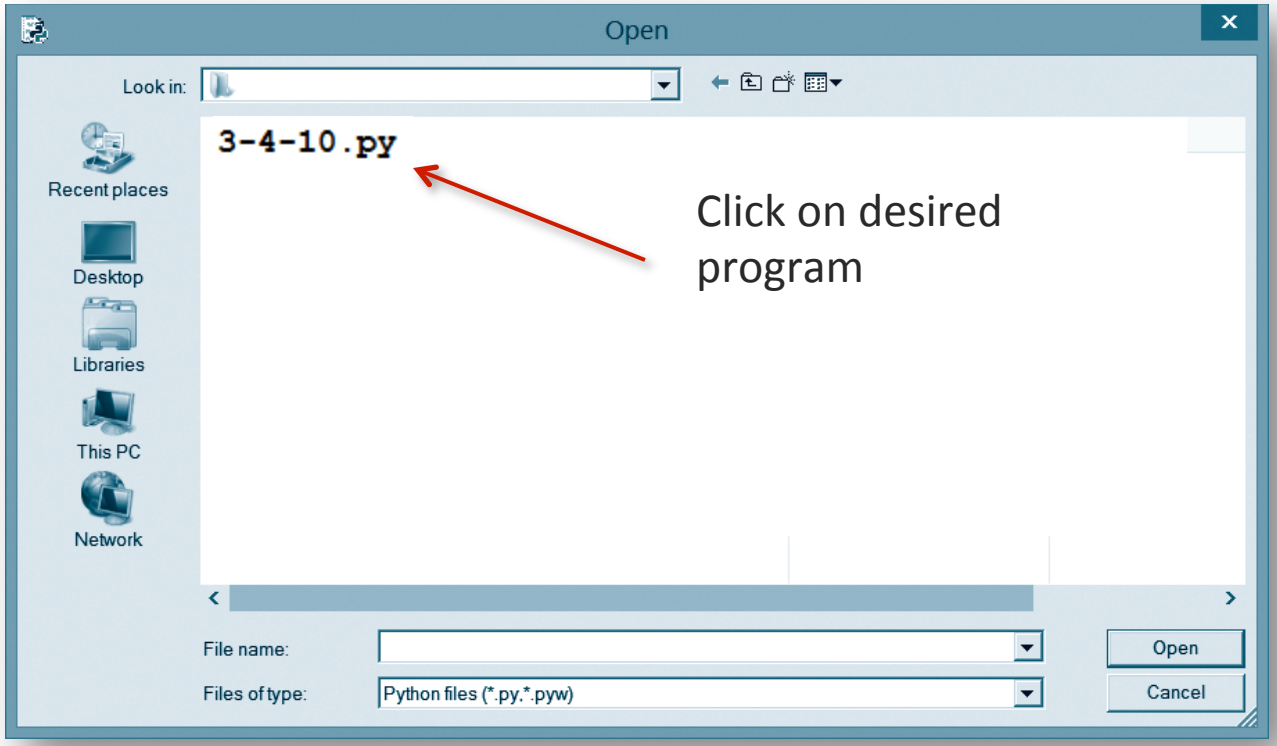
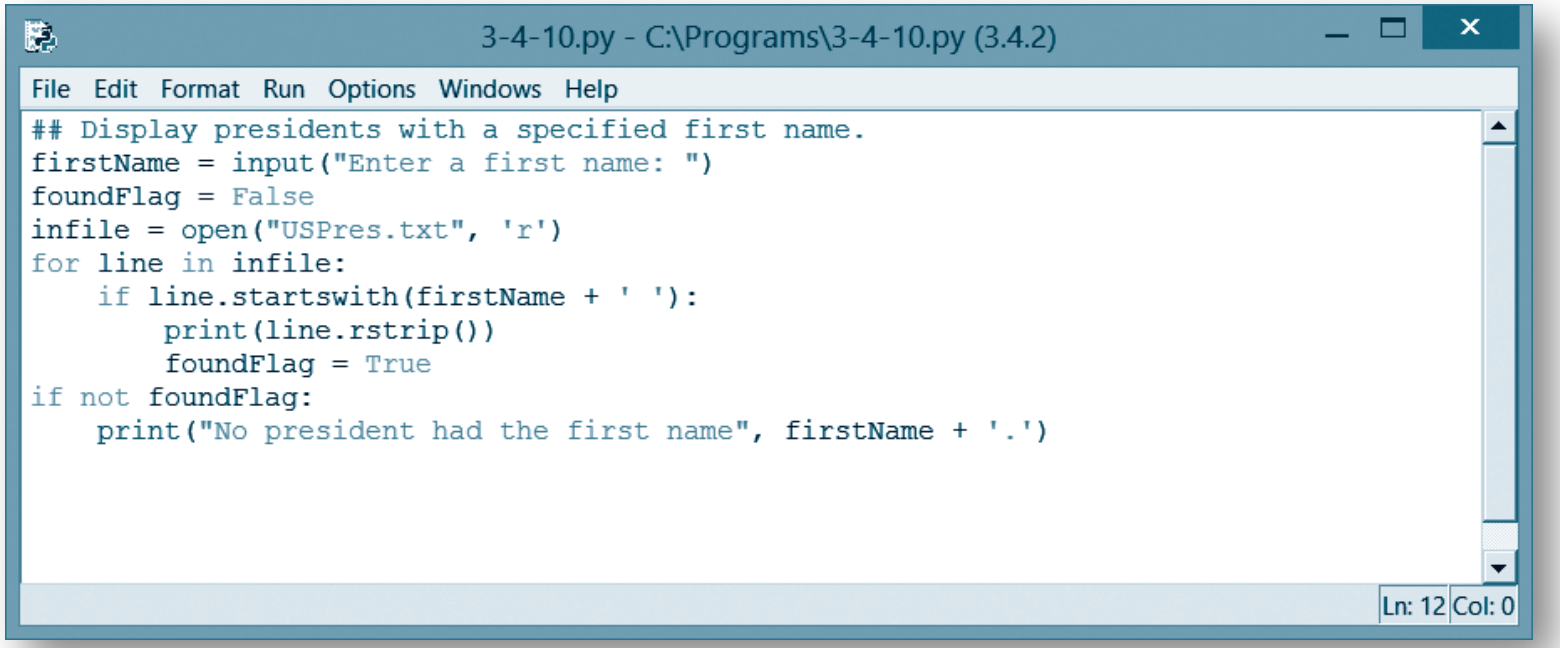


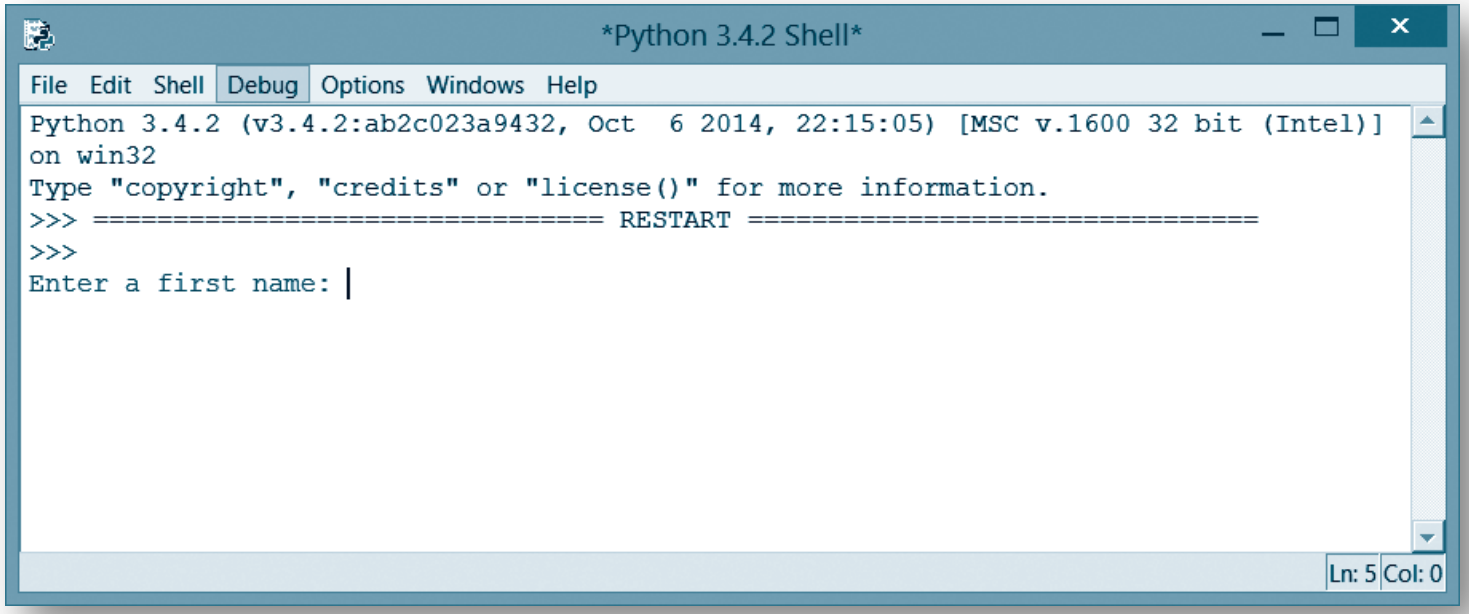
FIGURE 1.26 An Open dialog box.

An Open-a-Program Walkthrough



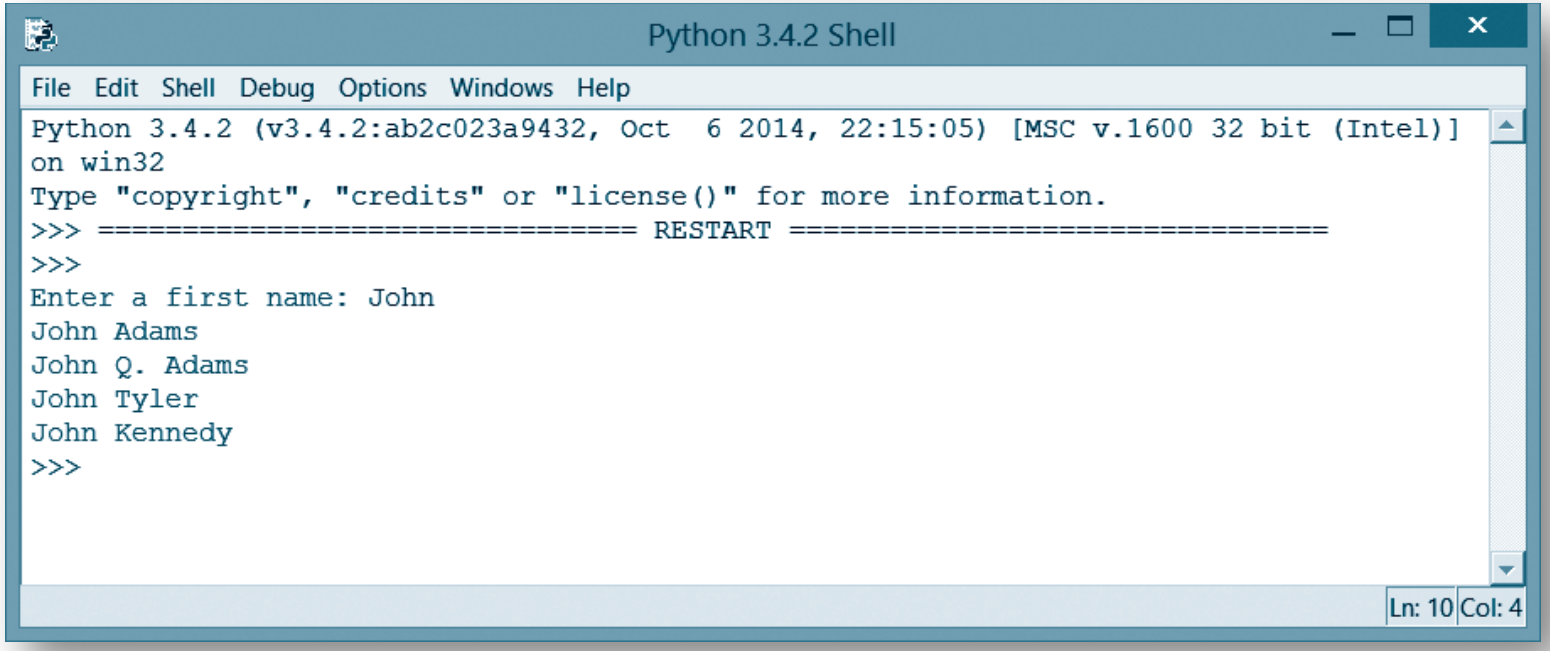
Example 10 of Section 3.4.

An Open-a-Program Walkthrough



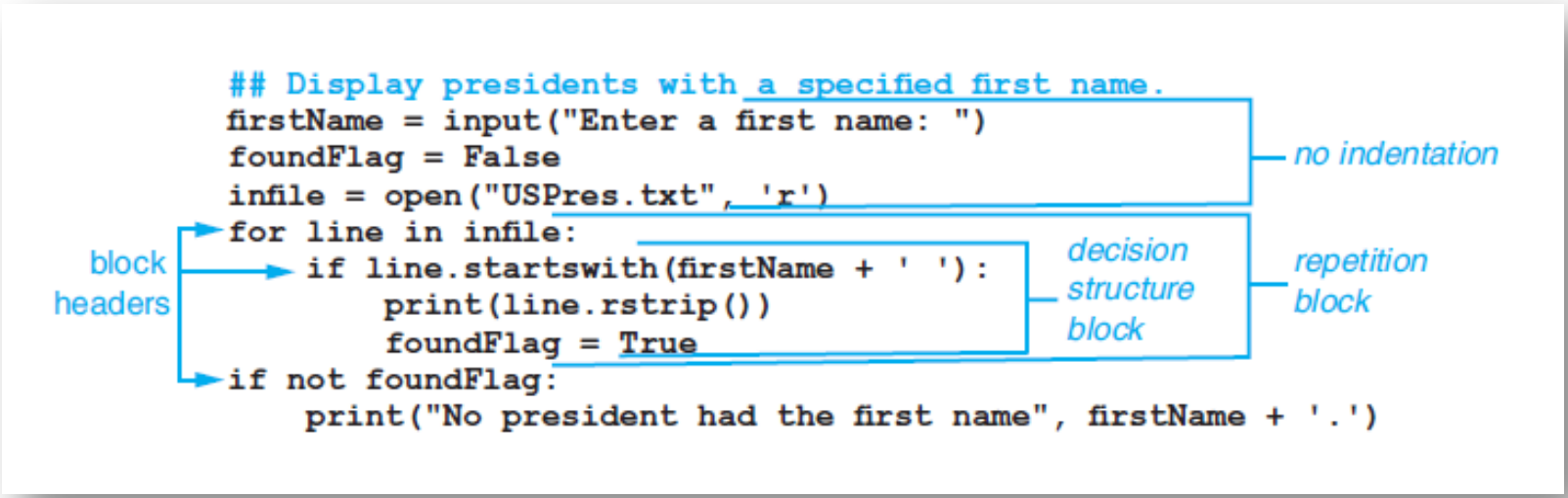
Request for input from Example 10 of Section 3.4.

An Open-a-Program Walkthrough



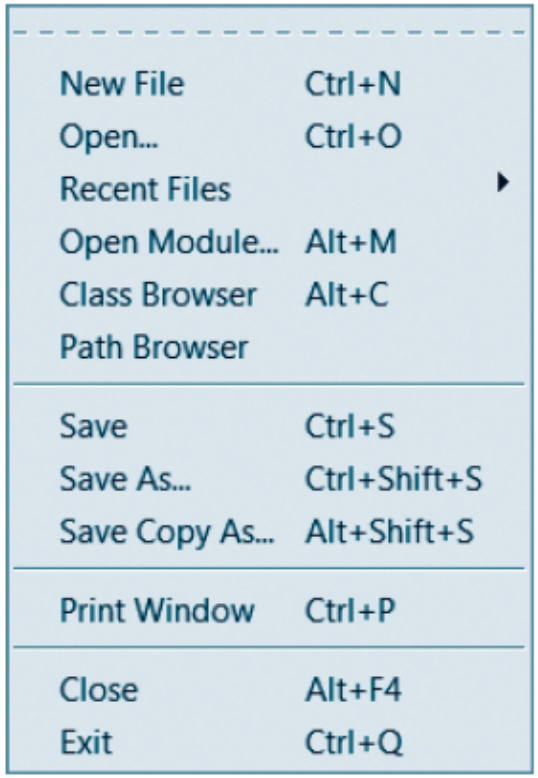
Complete output for Example 10 of Section 3.4.

Block-Structure



Example 10 of Section 3.4.

The File Drop-down Menu



SOME COMMANDS FROM THE *FILE* DROP-DOWN MENU.

COMMAND	EFFECT
New File	Create a new code editor window.
Open	Open a saved program.
Recent Files	Display a list of the most recently accessed programs.
Save	Save the current program.
Save As	Save the current program with a different name and possibly different location.
Print Window	Print a copy of the program on a printer.
Close	Close the current window.
Exit	Terminate Python.

The File drop-down menu.

Numbers

- Numbers are referred to as numeric literals
- Two Types of numbers: ints and floats
  - Whole number written without a decimal point called an int
  - Number written with a decimal point called a float

Numbers

- Arithmetic Operators
  - Addition, subtraction, multiplication, division, and exponentiation.
- Result of a division is always a float
- Result of the other operations is a float if …
  - Either of the numbers is a float
  - Otherwise is an int.

The print Function

- Used to display numbers on the monitor
  - If n is a number, print(n) displays number n.
  - The print function can display the result of evaluated expressions
  - A single print function can display several values
- Example 1: Program demonstrates operations

```
print(3 + 2, 3 - 2, 3 * 2)
print(8 / 2, 8 ** 2, 2 * (3 + 4))
```

[Run]

```
5 1 6
4.0 64 24
```

Variables

- In mathematics problems, quantities are referred to by names
- Names given to the values are called variables
- Example 2: Program uses speed, time … calculates distance

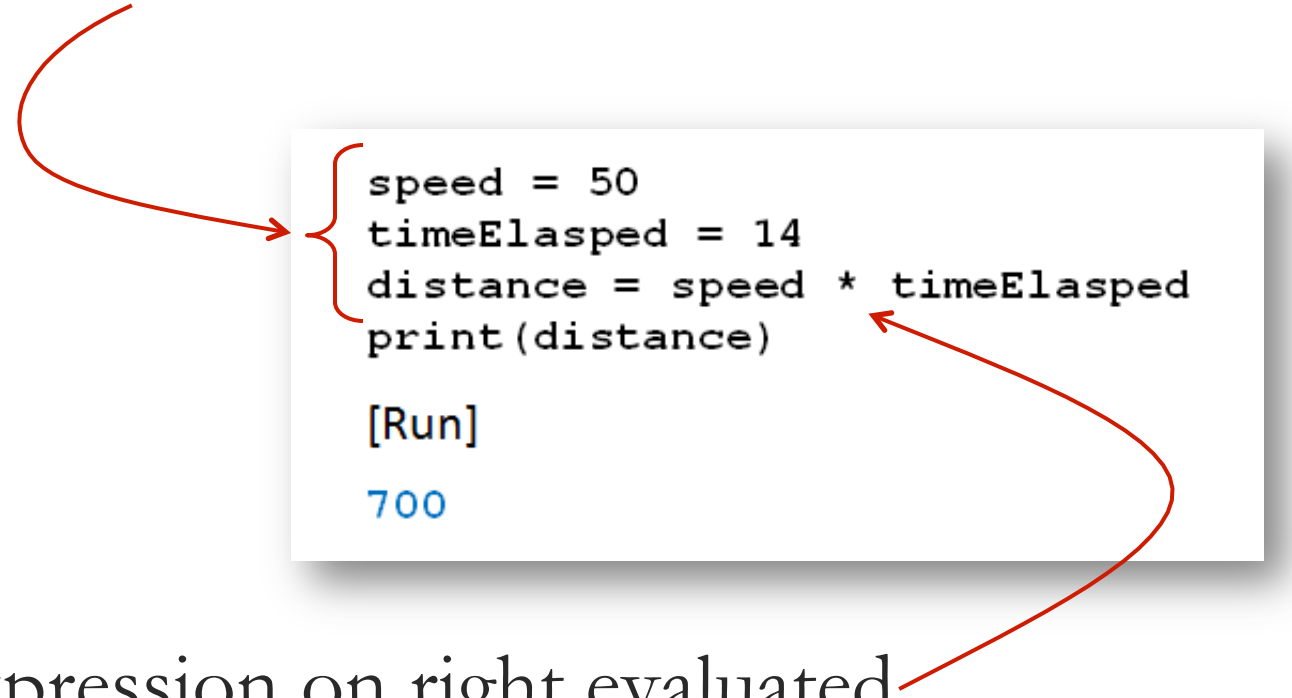
```
speed = 50
timeElasped = 14
distance = speed * timeElasped
print(distance)
```

[Run]

700

Variables

- Assignment statements



- Expression on right evaluated
  - That value assigned to variable

Variables

- Variable names in Python
  - Begin with letter or underscore _
  - Can only consist of letters, numbers, underscores
- Recommend using descriptive variable names
- Convention will be
  - Begin with lowercase
  - Use cap for additional "word" in the name
  - Example: rateOfChange

Variables

- Names in Python are case-sensitive
- There are thirty-three words, called reserved words (or keywords)
  - Have special meanings in Python
  - Cannot be used as variable names
  - See Appendix B

The abs Function

Expression	Value
abs(3)	3
abs(0)	0
abs(-3)	3

The int Function

Expression	Value
int(2.7)	2
int(3)	3
int(-2.7)	-2

The round Function

Expression	Value
round(2.7)	3
round(2.317, 2)	2.32
round(2.317, 1)	2.3

abs, int, and round Example

- Example 3: Program evaluates functions, prints results

```
a = 2
b = 3
print(abs(1 - (4 * b)))
print(int((a ** b) + .8))
print(round(a / b, 3))
```

[Run]

```
11
8
0.667
```

Augmented Assignments

- Remember: expression on right side of assignment statement evaluated before assignment is made

  var = var + 1

- Python has special operator to accomplish same thing

  var += 1

Augmented Assignments

- Example 4: Program illustrates different augmented assignment operators.

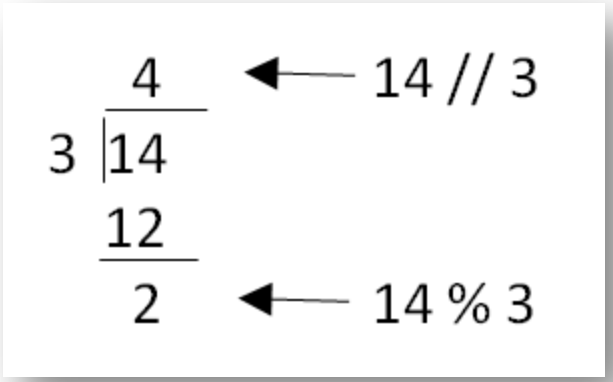
```
num1 = 6
num1 += 1
num2 = 7
num2 -= 5
num3 = 8
num3 /= 2
print(num1, num2, round(num3))
num1 = 1
num1 *= 3
num2 = 2
num2 **= 3
print(num1, num2)
```

[Run]

```
7 2 4
3 8
```

Two Other Integer Operators

- Integer division operator
  - Written //
- Modulus operator
  - Written %



Two Other Integer Operators

- Example 5: converts 41 inches to 3 feet, 5 inches:

```
totalInches = 41
feet = totalInches // 12
inches = totalInches % 12
print(feet, inches)
```

[Run]

```
3 5
```

Parentheses, Order of Precedence

- Parentheses used to clarify meaning of an expression
- Order of precedence
  - Terms inside parentheses (inner to outer)
  - Exponentiation
  - Multiplication, division (ordinary and integer), modulus
  - Addition and subtraction

Syntax Errors

- Grammatical and punctuation errors are called syntax errors.

Statement	*Reason for Error*
`print(3))`	The statement contains an extraneous right parenthesis.
`for = 5`	A reserved word was used as a variable name.
`print(2; 3)`	The semicolon should be a comma.

Table 2.1 Three Syntax Errors

Syntax Errors

FIGURE 2.1 Syntax error message boxes.

Runtime Errors

- Errors discovered while program is running called runtime errors or exceptions

Statement	*Reason for Error*
`primt(5)`	The function print is misspelled.
`x += 1`, when *x* has not been created	Python is not aware of the variable *x*.
`print(5 / 0)`	A number cannot be divided by zero.

Table 2.2 Three Runtime Errors

Runtime Errors

- When Python encounters exception
  - Terminates execution of program, displays message

```
Traceback (most recent call last):
  File "C:\test1.py", line 2, in <module>
    print(5 / 0) #ZeroDivisionError
ZeroDivisionError: division by zero
```

FIGURE 2.2 An Error Message for an Exception Error

Logic Error

- Occurs when a program does not perform the way it was intended
- Example
  average = firstNum + secondNum / 2
- Syntax correct, logic wrong: should be
  average = (firstNum + secondNum) / 2
- Logic errors are most difficult type of error to locate

Numeric Objects in Memory

- Consider this code:

```
n = 5
n = 7
```

- This is what happens:

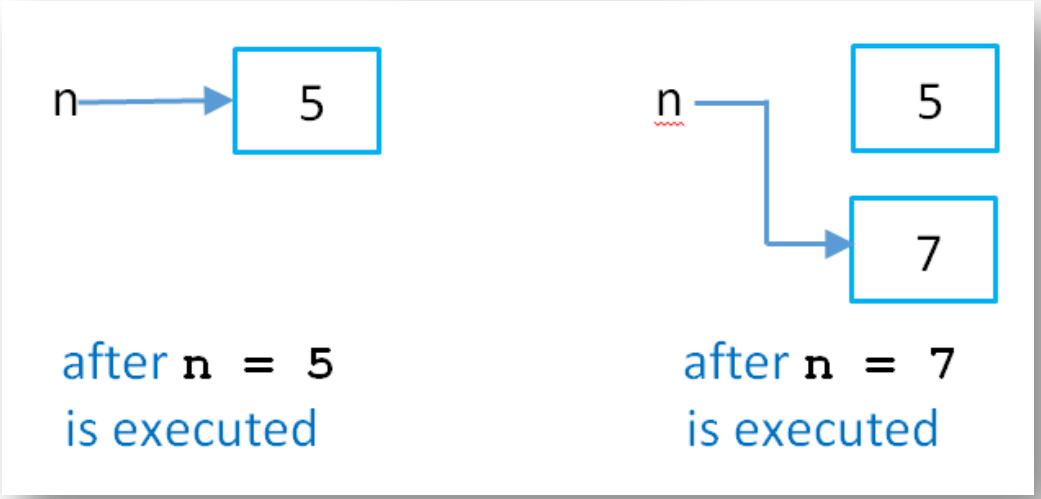


FIGURE 2.3 Numeric objects in memory.

Questions

- Determine the output of the following pieces of code:

```
a = 3
b = 5
print(a * b ** 2)
```

```
d = 5
d -= 1
print(d, d + 1, d - 2)
```

```
n = 5
n ** = 2
print(n/5)
```

```
points = 30
points += 20 * 10
print(points)
```

```
totalBerries = 100
totalCost = 352
eachBerry = totalCost /
            totalBerries
print(eachBerry)
```

```
totalMeters = 30255
kiloMeters = totalMeters // 1000
meters = totalMeters % 1000
print(kiloMeters, meters)
```

Questions

- Identify the errors:

```
a = 2
b = 3
a + b = c
print(b)
```

```
balance = 1,234
deposit = $100
print(Balance + Deposit)
```

```
0.05 = interest
balance = 800
print(interest * balance)
```

```
9W = 2 * 9W
print(9W)
```

Time to code - Exercise 1

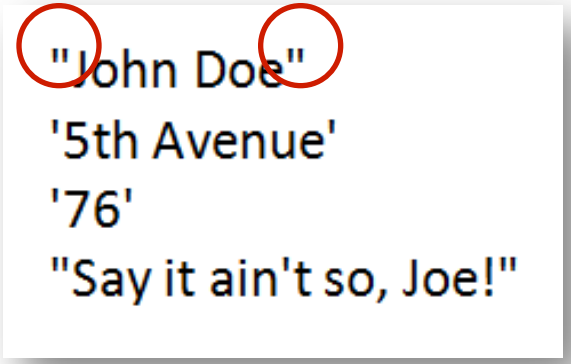
- Stock Purchase - The following steps calculate the amount of a stock purchase.
  a) Create the variable costPerShare and assign it the value 25.625.
  b) Create the variable numberOfShares and assign it the value 400
  c) Create the variable amount and assign it the product of the values of costPerShare and numberOfShares.
  d) Display the value of the variable amount.

Time to code – Exercise 2

- Discounted Price - The following steps calculate the price of an item after a 30% reduction.
  a) Create the variable price and assign it the value 19.95.
  b) Create the variable discountPercent and assign it the value 30.
  c) Create the variable markdown and assign it the value of (discountPercent divided by 100) times the value of price.
  d) Decrease the value of price by markdown.
  e) Display the value of price (rounded to two decimal places).

String Literals

- Sequence of characters that is treated as a single item
- Written as a sequence of characters surrounded by either single quotes (') or double quotes (").



Opening and closing quotation marks must be the same type

String Variables

- Variables also can be assigned string values
- Created (come into existence) the first time they appear in assignment statements
- When an argument of a print statement
  - Quotation marks not included in display

Indices and Slices

- Position or index of a character in a string
  - Identified with one of the numbers 0, 1, 2, 3, ...

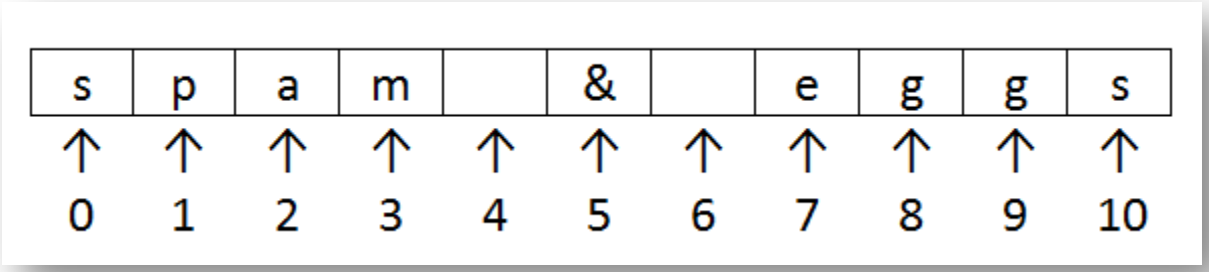


FIGURE 2.4 Indices of the characters of the string "spam & eggs".

Indices and Slices

- If str1 is a string, then str1[m:n] is the substring beginning at position m and ending at position n - 1
  - Example "spam & eggs"[2:7]

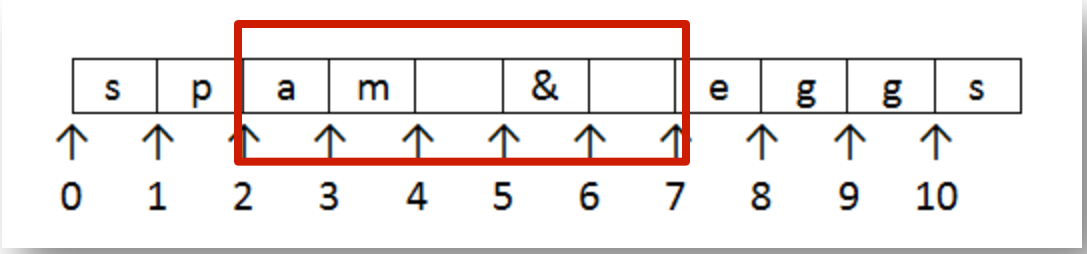


FIGURE 2.5 Aid to visualizing slices.

Indices and Slices

- Example 1: Program shows use of indices

```
print("Python")
print("Python"[1], "Python"[5], "Python"[2:4])
str1 = "Hello World!"
print(str1.find('W'))
print(str1.find('x'))
print(str1.rfind('l'))
```

[Run]

```
Python
y n th
6
-1
9
```

Negative Indices

- Python allows strings to be indexed by their position with regards to the right
  - Use negative numbers for indices.

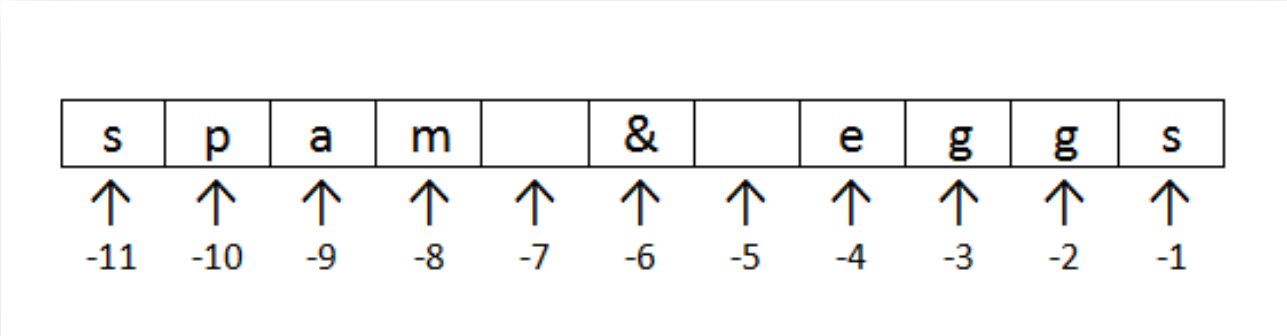


FIGURE 2.6 Negative indices of the characters of the string "spam & eggs".

Negative Indices

- Example 2: Program illustrates negative indices.

```
print("Python")
print("Python"[-1], "Python"[-4], "Python"[-5:-2])
str1 = "spam & eggs"
print(str1[-2])
print(str1[-8:-3])
print(str1[0:-1])
```

[Run]

```
Python
n t yth
g
m & e
spam & egg
```

Default Bounds for Slices

- Example 3: Program illustrates default bounds

```
print("Python"[2:], "Python"[:4], "Python"[:])
print("Python"[-3:], "Python"[:-3])
```

[Run]

```
thon Pyth Python
hon Pyt
```

String Concatenation

- Two strings can be combined to form a new string
  - Consisting of the strings joined together
  - Represented by a plus sign
- Combination of strings, plus signs, functions, and methods can be evaluated
  - Called a string expression

String Repetition

- Asterisk operator can be used with strings to repeatedly concatenate a string with itself

Expression	Value
"ha" * 4	"hahahaha"
"mur" * 2	"murmur"
'x' * 10	"xxxxxxxxxx"
("cha-" * 2) + "cha"	"cha-cha-cha"

String Functions and Methods

Function or Method	Example	Value	Description
len	len(str1)	6	number of characters in the string
upper	str1.upper()	"PYTHON"	uppercases every alphabetical character
lower	str1.lower()	"python"	lowercases every alphabetical character
count	str1.count('th')	1	number of non-overlapping occurrences of the substring
capitalize	"coDE".capitalize()	"Code"	capitalizes the first letter of the string and lowercases the rest
title	"beN hur".title()	"Ben Hur"	capitalizes the first letter of each word in the string and lowercases the rest
rstrip	"ab ".rstrip()	"ab"	removes spaces from the right side of the string

Table 2.3 String Operations (str1 = "Python")

Time to code – Exercise 3

- Inventor - The following steps give the name and birth year of a famous inventor.
  a) Create the variable firstName and assign it the value "Thomas".
  b) Create the variable middleName and assign it the value "Alva".
  c) Create the variable lastName and assign it the value "Edison".
  d) Create the variable yearOfBirth and assign it the value 1847.
  e) Display the phrase "The year of birth of" followed by the inventor's full name, followed by "is", and the inventor's year of birth.

Questions ?

Computer Organization

- Regardless of differences in physical appearance, computers can be envisioned as divided into various logical units or sections.

Logical unit	Description
Input unit	This "receiving" section obtains information (data and computer programs) from input devices and places it at the disposal of the other units for processing. Most information is entered into computers through keyboards, touch screens and mouse devices. Other forms of input include receiving voice commands, scanning images and barcodes, reading from secondary storage devices (such as hard drives, DVD drives, Blu-ray Disc™ drives and USB flash drives—also called "thumb drives" or "memory sticks"), receiving video from a webcam and having your computer receive information from the Internet (such as when you download videos from YouTube™ or e-books from Amazon). Newer forms of input include position data from a GPS device, and motion and orientation information from an accelerometer in a smartphone or game controller (such as Microsoft® Kinect™, Wii™ Remote and PlayStation® Move).

Computer Organisation

Logical unit	Description
Output unit	This "shipping" section takes information that the computer has processed and places it on various output devices to make it available for use outside the computer. Most information that's output from computers today is displayed on screens, printed on paper, played as audio or video on PCs and media players (such as Apple's popular iPods) and giant screens in sports stadiums, transmitted over the Internet or used to control other devices, such as robots and "intelligent" appliances.
Memory unit	This rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is *volatile*—it's typically lost when the computer's power is turned off. The memory unit is often called either memory or primary memory. Typical main memories on desktop and notebook computers contain between 1 and 8 GB (GB stands for gigabytes; a gigabyte is approximately one billion bytes).

Computer Organisation

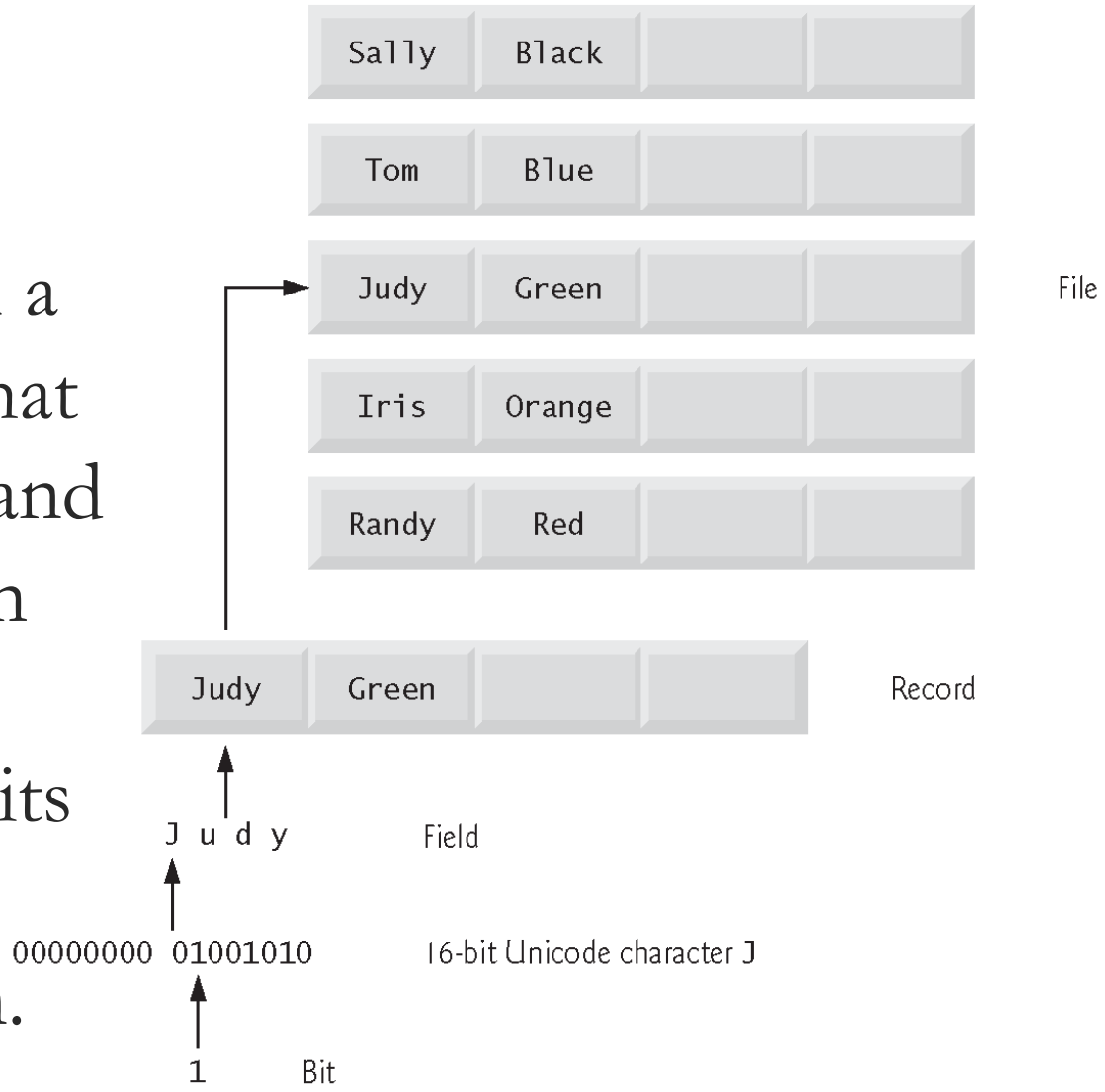
Logical unit	Description
Arithmetic and logic unit (ALU)	This "manufacturing" section performs *calculations*, such as addition, subtraction, multiplication and division. It also contains the *decision* mechanisms that allow the computer, for example, to compare two items from the memory unit to determine whether they're equal. In today's systems, the ALU is usually implemented as part of the next logical unit, the CPU.
Central processing unit (CPU)	This "administrative" section coordinates and supervises the operation of the other sections. The CPU tells the input unit when information should be read into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today's computers have multiple CPUs and, hence, can perform many operations simultaneously. A multi-core processor implements multiple processors on a single integrated-circuit chip—a *dual-core processor* has two CPUs and a *quad-core processor* has four CPUs. Today's desktop computers have processors that can execute billions of instructions per second.

Computer Organisation

Logical unit	Description
Secondary storage unit	This is the long-term, high-capacity "warehousing" section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your *hard drive*) until they're again needed, possibly hours, days, months or even years later. Information on secondary storage devices is *persistent*—it's preserved even when the computer's power is turned off. Secondary storage information takes much longer to access than information in primary memory, but the cost per unit of secondary storage is much less than that of primary memory. Examples of secondary storage devices include CD drives, DVD drives and flash drives, some of which can hold up to 512 GB. Typical hard drives on desktop and notebook computers can hold up to 2 TB (TB stands for terabytes; a terabyte is approximately one trillion bytes).

Data Hierarchy

- Data items processed by computers form a data hierarchy that becomes larger and more complex in structure as we progress from bits to characters to fields, and so on.



Levels of Data Hierarchy

Level	Description
Bits	The smallest data item in a computer can assume the value 0 or the value 1. Such a data item is called a bit (short for "binary digit"—a digit that can assume one of two values). It's remarkable that the impressive functions performed by computers involve only the simplest manipulations of 0s and 1s—*examining a bit's value*, *setting a bit's value* and *reversing a bit's value* (from 1 to 0 or from 0 to 1).
Characters	It's tedious for people to work with data in the low-level form of bits. Instead, they prefer to work with *decimal digits* (0–9), *letters* (A–Z and a–z), and *special symbols* (e.g., $, @, %, &, *, (,), –, +, ", :, ? and /). Digits, letters and special symbols are known as characters. The computer's character set is the set of all the characters used to write programs and represent data items. Computers process only 1s and 0s, so every character is represented as a pattern of 1s and 0s. The Unicode character set contains characters for many of the world's languages. C supports several character sets, including 16-bit Unicode® characters

Levels of Data Hierarchy

Level	Description
Characters (cont.)	that are composed of two bytes, each composed of eight bits. See Appendix B for more information on the ASCII (American Standard Code for Information Interchange) character set—the popular subset of Unicode that represents uppercase and lowercase letters, digits and some common special characters.
Fields	Just as characters are composed of bits, fields are composed of characters or bytes. A field is a group of characters or bytes that conveys meaning. For example, a field consisting of uppercase and lowercase letters could be used to represent a person's name, and a field consisting of decimal digits could represent a person's age.

Levels of Data Hierarchy

Level	Description
Records	Several related fields can be used to compose a record. In a payroll system, for example, the record for an employee might consist of the following fields (possible types for these fields are shown in parentheses): • Employee identification number (a whole number) • Name (a string of characters) • Address (a string of characters) • Hourly pay rate (a number with a decimal point) • Year-to-date earnings (a number with a decimal point) • Amount of taxes withheld (a number with a decimal point) Thus, a record is a group of related fields. In the preceding example, all the fields belong to the same employee. A company might have many employees and a payroll record for each one.

Levels of Data Hierarchy

Level	Description
Files	A file is a group of related records. [*Note:* More generally, a file contains arbitrary data in arbitrary formats. In some operating systems, a file is viewed simply as a *sequence of bytes*—any organization of the bytes in a file, such as organizing the data into records, is a view created by the application programmer.] It's not unusual for an organization to have many files, some containing billions, or even trillions, of characters of information.
Database	A database is an electronic collection of data that's organized for easy access and manipulation. The most popular database model is the relational database in which data is stored in simple *tables*. A table includes *records* and *fields*. For example, a table of students might include first name, last name, major, year, student ID number and grade point average. The data for each student is a record, and the individual pieces of information in each record are the fields. You can search, sort and manipulate the data based on its relationship to multiple tables or databases. For example, a university might use data from the student database in combination with databases of courses, on-campus housing, meal plans, etc.