

Topics

- Designing a Program
- Input, Processing, and Output
- Displaying Output with `print` Function
- Comments
- Variables
- Reading Input from the Keyboard
- Performing Calculations
- More About Data Output

Designing a Program

- Programs must be designed before they are written
- Program development cycle:
 - Design the program
 - Write the code
 - Correct syntax errors
 - Test the program
 - Correct logic errors

Designing a Program (cont'd.)

- Design is the most important part of the program development cycle
- Understand the task that the program is to perform
 - Work with customer to get a sense what the program is supposed to do
 - Ask questions about program details
 - Create one or more software requirements

Designing a Program (cont'd.)

- Determine the steps that must be taken to perform the task
 - Break down required task into a series of steps
 - Create an algorithm, listing logical steps that must be taken
- **Algorithm:** set of well-defined logical steps that must be taken to perform a task

Pseudocode

- **Pseudocode:** fake code
 - Informal language that has no syntax rule
 - Not meant to be compiled or executed
 - Used to create model program
 - No need to worry about syntax errors, can focus on program's design
 - Can be translated directly into actual code in any programming language

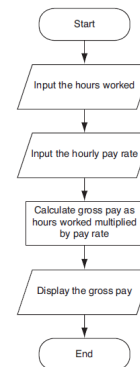
Flowcharts

- **Flowchart:** diagram that graphically depicts the steps in a program
- Ovals are terminal symbols
- Parallelograms are input and output symbols
- Rectangles are processing symbols
- Symbols are connected by arrows that represent the flow of the program

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Figure 2-2 Flowchart for the pay calculating program



Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Input, Processing, and Output

- Typically, computer performs three-step process
- Receive input
 - Input: any data that the program receives while it is running
- Perform some process on the input
 - Example: mathematical calculation
- Produce output

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Displaying Output with the print Function

- **Function:** piece of prewritten code that performs an operation
- **print function:** displays output on the screen
- **Argument:** data given to a function
 - Example: data that is printed to screen
- **Statements in a program execute in the order that they appear**
 - From top to bottom

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Strings and String Literals

- **String:** sequence of characters that is used as data
- **String literal:** string that appears in actual code of a program
 - Must be enclosed in single (') or double (") quote marks
 - String literal can be enclosed in triple quotes (''' or """)
 - Enclosed string can contain both single and double quotes and can have multiple lines

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Comments

- **Comments:** notes of explanation within a program
 - Ignored by Python interpreter
 - Intended for a person reading the program's code
 - Begin with a # character
- **End-line comment:** appears at the end of a line of code
 - Typically explains the purpose of that line

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Variables

- **Variable:** name that represents a value stored in the computer memory
 - Used to access and manipulate data stored in memory
 - A variable references the value it represents
- **Assignment statement:** used to create a variable and make it reference data
 - General format is `variable = expression`
 - Example: `age = 29`
 - **Assignment operator:** the equal sign (=)

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Variables (cont'd.)

- In assignment statement, variable receiving value must be on left side
- A variable can be passed as an argument to a function
 - Variable name should not be enclosed in quote marks
- You can only use a variable if a value is assigned to it

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Variable Naming Rules

- **Rules for naming variables in Python:**
 - Variable name cannot be a Python key word
 - Variable name cannot contain spaces
 - First character must be a letter or an underscore
 - After first character may use letters, digits, or underscores
 - Variable names are case sensitive
- **Variable name should reflect its use**

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Displaying Multiple Items with the `print` Function

- Python allows one to display multiple items with a single call to `print`
 - Items are separated by commas when passed as arguments
 - Arguments displayed in the order they are passed to the function
 - Items are automatically separated by a space when displayed on screen

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Variable Reassignment

- Variables can reference different values while program is running
- **Garbage collection:** removal of values that are no longer referenced by variables
 - Carried out by Python interpreter
- A variable can refer to item of any type
 - Variable that has been assigned to one type can be reassigned to another type

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Numeric Data Types, Literals, and the `str` Data Type

- **Data types:** categorize value in memory
 - e.g., `int` for integer, `float` for real number, `str` used for storing strings in memory
- **Numeric literal:** number written in a program
 - No decimal point considered `int`, otherwise, considered `float`
- Some operations behave differently depending on data type

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

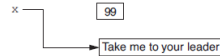
Reassigning a Variable to a Different Type

- A variable in Python can refer to items of any type

Figure 2-7 The variable `x` references an integer



Figure 2-8 The variable `x` references a string



Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Reading Input from the Keyboard

- Most programs need to read input from the user
- Built-in `input` function reads input from keyboard
 - Returns the data as a string
 - Format: `variable = input(prompt)`
 - `prompt` is typically a string instructing user to enter a value
 - Does not automatically display a space after the prompt

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Reading Numbers with the `input` Function

- `input` function always returns a string
- Built-in functions convert between data types
 - `int(item)` converts `item` to an `int`
 - `float(item)` converts `item` to a `float`
 - Nested function call: general format:
`function1(function2(argument))`
 - value returned by `function2` is passed to `function1`
 - Type conversion only works if item is valid numeric value, otherwise, throws exception

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Performing Calculations

- Math expression: performs calculation and gives a value
 - Math operator: tool for performing calculation
 - Operands: values surrounding operator
 - Variables can be used as operands
 - Resulting value typically assigned to variable
- Two types of division:
 - `/` operator performs floating point division
 - `//` operator performs integer division
 - Positive results truncated, negative rounded away from zero

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Operator Precedence and Grouping with Parentheses

- Python operator precedence:
 - Operations enclosed in parentheses
 - Forces operations to be performed before others
 - Exponentiation (`**`)
 - Multiplication (`*`), division (`/` and `//`), and remainder (`%`)
 - Addition (`+`) and subtraction (`-`)
- Higher precedence performed first
 - Same precedence operators execute from left to right

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

The Exponent Operator and the Remainder Operator

- Exponent operator (`**`): Raises a number to a power
 - `x ** y = xy`
- Remainder operator (`%`): Performs division and returns the remainder
 - a.k.a. modulus operator
 - e.g., `4%2=0`, `5%2=1`
 - Typically used to convert times and distances, and to detect odd or even numbers

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Converting Math Formulas to Programming Statements

- Operator required for any mathematical operation
- When converting mathematical expression to programming statement:
 - May need to add multiplication operators
 - May need to insert parentheses

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Mixed-Type Expressions and Data Type Conversion

- Data type resulting from math operation depends on data types of operands
 - Two `int` values: result is an `int`
 - Two `float` values: result is a `float`
 - `int` and `float`: `int` temporarily converted to `float`, result of the operation is a `float`
 - Mixed-type expression
 - Type conversion of `float` to `int` causes truncation of fractional part

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Breaking Long Statements into Multiple Lines

- Long statements cannot be viewed on screen without scrolling and cannot be printed without cutting off
- Multiline continuation character (\): Allows to break a statement into multiple lines

• Example:

```
print('my first name is', \
      first_name)
```

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

More About Data Output

- `print` function displays line of output
 - Newline character at end of printed data
 - Special argument `end='delimiter'` causes `print` to place `delimiter` at end of data instead of newline character
- `print` function uses space as item separator
 - Special argument `sep='delimiter'` causes `print` to use `delimiter` as item separator

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

More About Data Output (cont'd.)

- Special characters appearing in string literal
 - Preceded by backslash (\)
 - Examples: newline (`\n`), horizontal tab (`\t`)
 - Treated as commands embedded in string
- When `+` operator used on two strings in performs string concatenation
 - Useful for breaking up a long string literal

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Formatting Numbers

- Can format display of numbers on screen using built-in `format` function
 - Two arguments:
 - Numeric value to be formatted
 - Format specifier
 - Returns string containing formatted number
 - Format specifier typically includes precision and data type
 - Can be used to indicate scientific notation, comma separators, and the minimum field width used to display the value

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Formatting Numbers (cont'd.)

- The % symbol can be used in the format string of `format` function to format number as percentage
- To format an integer using `format` function:
 - Use `d` as the type designator
 - Do not specify precision
 - Can still use `format` function to set field width or comma separator

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

Summary

- This chapter covered:
 - The program development cycle, tools for program design, and the design process
 - Ways in which programs can receive input, particularly from the keyboard
 - Ways in which programs can present and format output
 - Use of comments in programs
 - Uses of variables
 - Tools for performing calculations in programs

Addison-Wesley
is an imprint of
PEARSON

Copyright © 2015 Pearson Education, Inc. Publishing as Pearson Addison-Wesley