TEC222 Introduction to Programming with Python

Lecture 1

Introduction

Variables and Types

About me

- Instructor: Kosmas O. Kosmopoulos
- Email: kosmas.kosmopoulos@faculty.hult.edu
- Office Hours:

  Office hours will be held on Tuesday / Thursday 12:00-13:00 and Wednesday 17:30-18:00

Introduction to the course

- This is an introductory course in Python programming for the students with little or no programming experience.
- Python is a language with a simple syntax, and a powerful set of libraries. It is an interpreted language, with a rich programming environment, including a robust debugger and profiler.
- While it is easy for beginners to learn, it is widely used and has many interdisciplinary applications
- The course introduces students to problem-solving methods, algorithm development, data types, control flow, object-oriented programming, and graphical user interface-driven applications, via the Python programming language .

What we will do...

- You will solve problems, explore real-world software development challenges, and create practical and contemporary applications.
- You will write programs that are testable and maintainable (using good programming style, naming conventions, and comments).
- Topics include: Algorithms and and Problem Solving, Variables Input and Output, Numeric Data Types and Strings, Control Structures and Boolean logic, Functions and Program Design, Processing Data, Exception Handling, Turtle Graphics, Objects and Classes and Graphical User Interfaces (GUI)

...and how...

- The course is divided into theoretical-lecture and practical-laboratory/seminar parts.
  - Theoretical material will be delivered during the first half of each session.
  - Supervised laboratory and seminar sessions will be used to develop implementation skills understanding of the practical/programming concepts covered.
- **You will need to bring your laptops to every session.**

Recommended book

- **ESSENTIAL READING**

  Schneider, David I.(2016) *An Introduction to Programming Using Python, Global Edition*. Pearson (Intl), ISBN 13: 978-1-292-10343-3

- **RECOMMENDED READING**

  Downey, Allen B. (2015) *Think Python*, O'Reilly.

  You can download it for free here: http://greenteapress.com/wp/think-python-2e/

Housekeeping issues

- This is a 3 hours session so one break of approximately 20 min will be scheduled within the slot
- Time within the 3 hour slots will be allocated for the group assignment
- Students are expected to conduct themselves in a reasonable manner when attending sessions.
- This includes: switching off all mobile phones, staying in class during lecture and seminar times.

Assignment 1 – 30%

- 2 In-class Tests
  - **Weighting (% of final grade):** 30% (15% each)
  - **Learning Outcome(s) Assessed: LO1, LO2**
  - **Description of Assignment:** Students will be given two in-class tests covering the topics up to week 4 and then week 5 to 9. The test will be mostly multiple choice and will be comprised of programming exercises and theory questions
  - **Grading Criteria (What constitutes a good assignment?):** Students will receive full credit for each question by writing the correct answer

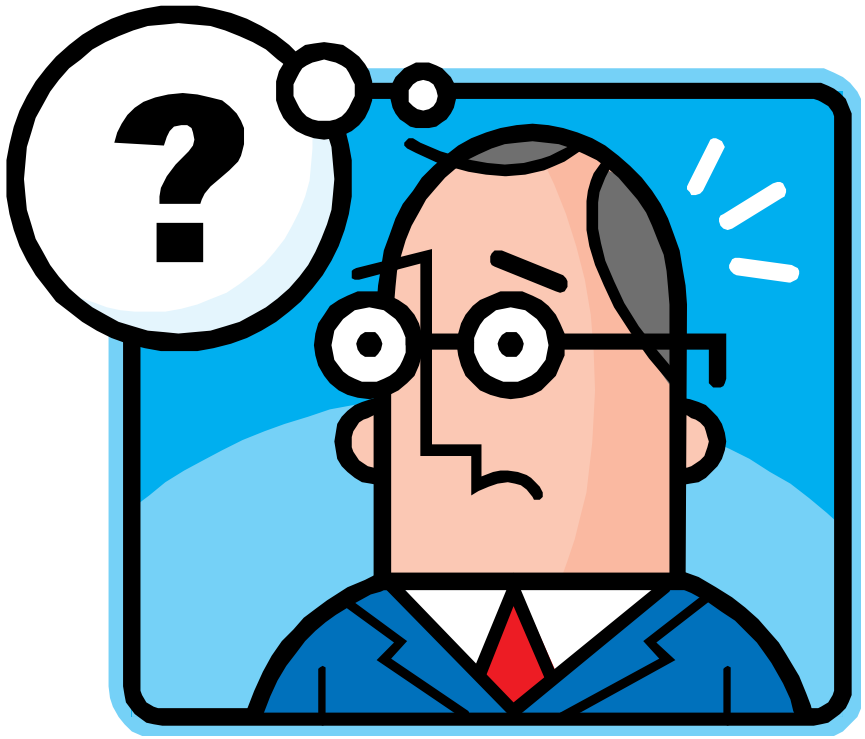
Assignment 2 – 20%

- Programming Portfolio
  - **Weighting (% of final grade):** 20%
  - **Learning Outcome(s) Assessed: LO1,LO2,LO3**
  - **Description of Assignment:** Students will be given small programming activities based on the topics covered, on every session starting from week 2
  - **Grading Criteria (What constitutes a good assignment?):** The tasks will be marked based on the completed program executing and fulfilling the required specification. The tasks would need to be submitted on time on order to achieve the full mark

Assignment 3 – 50%

- Programming project and Report
  - **Weighting (% of final grade):** 50% (Group Programming Project 30%, Individual Report 20%)
  - **Learning Outcome(s) Assessed: LO1,LO2,LO3**
  - **Description of Assignment:** Students will work in pairs towards a bigger programming project. Students will be requested to write an individual report (design notes) to reflect on the work that they and their partner have done on the programming project. The report should not exceed 1250 words
  - **Grading Criteria (What constitutes a good assignment?):** Students should submit a fully commented code that fulfills the coursework specifications. The code should run otherwise the mark would be capped at 60% for this component. They should also submit individually a report detailing how the project was done, which programming constructs were used, what they achieved and who has done which part of the work. Students might be asked to demo their work to the instructor.

Any questions?



Computers: Hardware and Software

- In use today are more than a billion general-purpose computers, and billions more embedded computers are used in cell phones, smartphones, tablet computers, home appliances, automobiles and more.
- Computers can perform computations and make logical decisions phenomenally faster than human beings can.
- Today's personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime.
- Supercomputers are already performing thousands of trillions (quadrillions) of instructions per second!
- Computers process data under the control of sets of instructions called computer programs.
- These programs guide the computer through ordered actions specified by people called computer programmers.

Computers: Hardware and Software

- The programs that run on a computer are referred to as software.
- A computer consists of various devices referred to as hardware
  - (e.g., the keyboard, screen, mouse, hard disks, memory, DVD drives and processing units).
- Computing costs are dropping dramatically, owing to rapid developments in hardware and software technologies.

Computers: Hardware and Software

- Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each.
- Silicon-chip technology has made computing so economical that more than computers have become a commodity.

Moore's Law

- For many decades, hardware costs have fallen rapidly.
- Every year or two, the capacities of computers have approximately doubled inexpensively.
- This remarkable trend often is called Moore's Law, named for the person who identified it, Gordon Moore, co-founder of Intel.

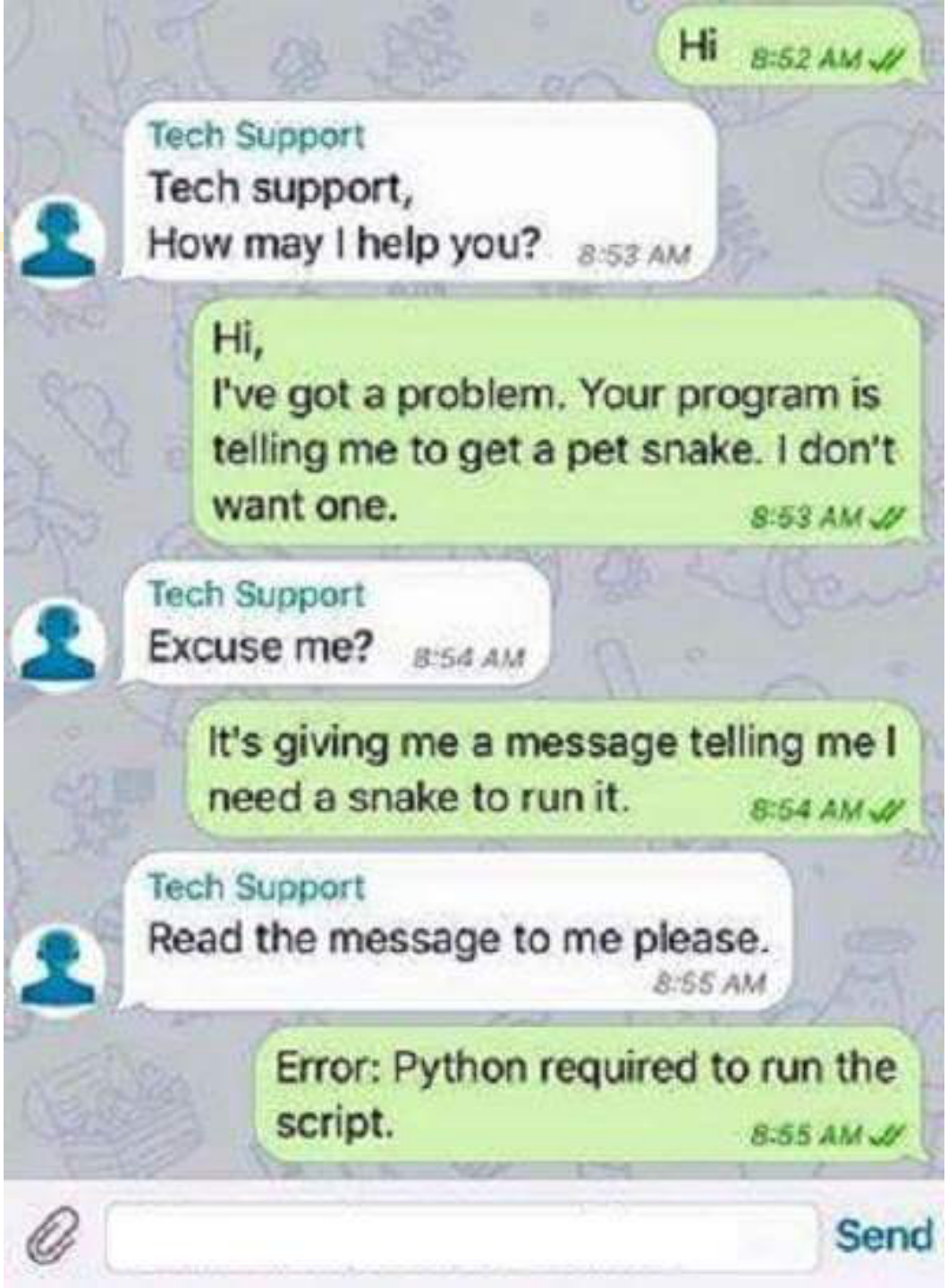
Computers: Hardware and Software

- Moore's Law and related observations apply especially to the amount of memory that computers have for programs, the amount of secondary storage (such as disk storage) they have to hold programs and data over longer periods of time, and their processor speeds—the speeds at which computers execute their programs (i.e., do their work).
- Similar growth has occurred in the communications field.

Computers: Hardware and Software

- Costs have plummeted as enormous demand for communications bandwidth (i.e., information-carrying capacity) has attracted intense competition.
- Such phenomenal improvement is fostering the Information Revolution.

Python


Hi 8:52 AM
Tech Support
Tech support,
How may I help you? 8:53 AM
Hi,
I've got a problem. Your program is telling me to get a pet snake. I don't want one. 8:53 AM
Tech Support
Excuse me? 8:54 AM
It's giving me a message telling me I need a snake to run it. 8:54 AM
Tech Support
Read the message to me please. 8:55 AM
Error: Python required to run the script. 8:55 AM
Send

Questions and Answers

- How do we communicate with the computer?
  - Programming languages
- How do we get computers to perform complicated tasks?
  - Tasks are broken down into a sequence of instructions
- Why Python?
  - Powerful, easy to download, write and read

Questions and Answers

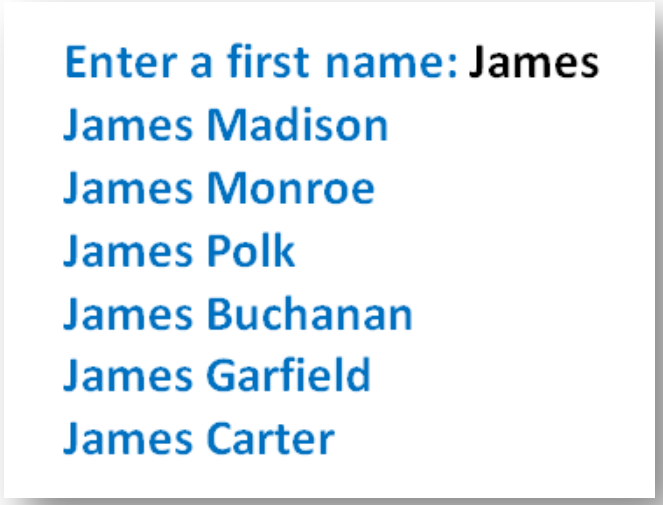
- How did the language Python get its name?
  - Named for the British comedy group Monty Python (really!)
- What is an interpreted language?
  - Uses an interpreter, translates high-level language one statement at a time into machine language and then runs

Questions and Answers

- How are problems solved with a program?
  - Step-by-step procedure devised to process given data and produce requested output.
- What is a zero-based numbering system?
  - Numbering begins with zero instead of one
- Prerequisites to learning Python?
  - Be familiar with how folders and files are managed

Questions and Answers

- What is an example of a program developed?

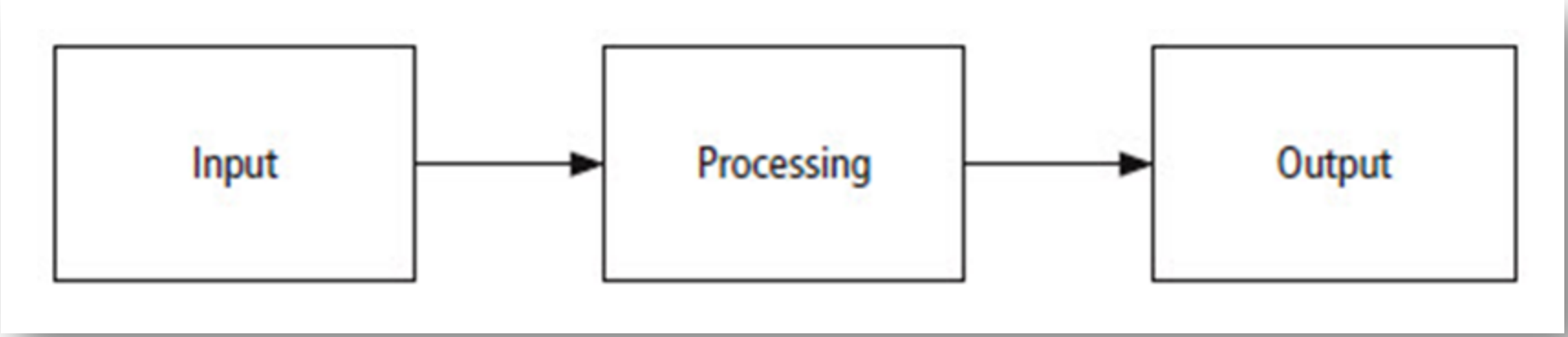


- How does the programmer create such a program?
  - About ten lines of code that search a text file and extract the requested names.

Questions and Answers

- Where can I research questions I have about Python?
  - Documentation at https://www.python.org/doc/

Program Development Cycle



The problem solving process.

Real Life Problem Solving

There are 4 steps to solving a problem:

UNDERSTAND THE PROBLEM
WORK OUT A PLAN
SORT OUT THE DETAILS
TEST AND EVALUATE

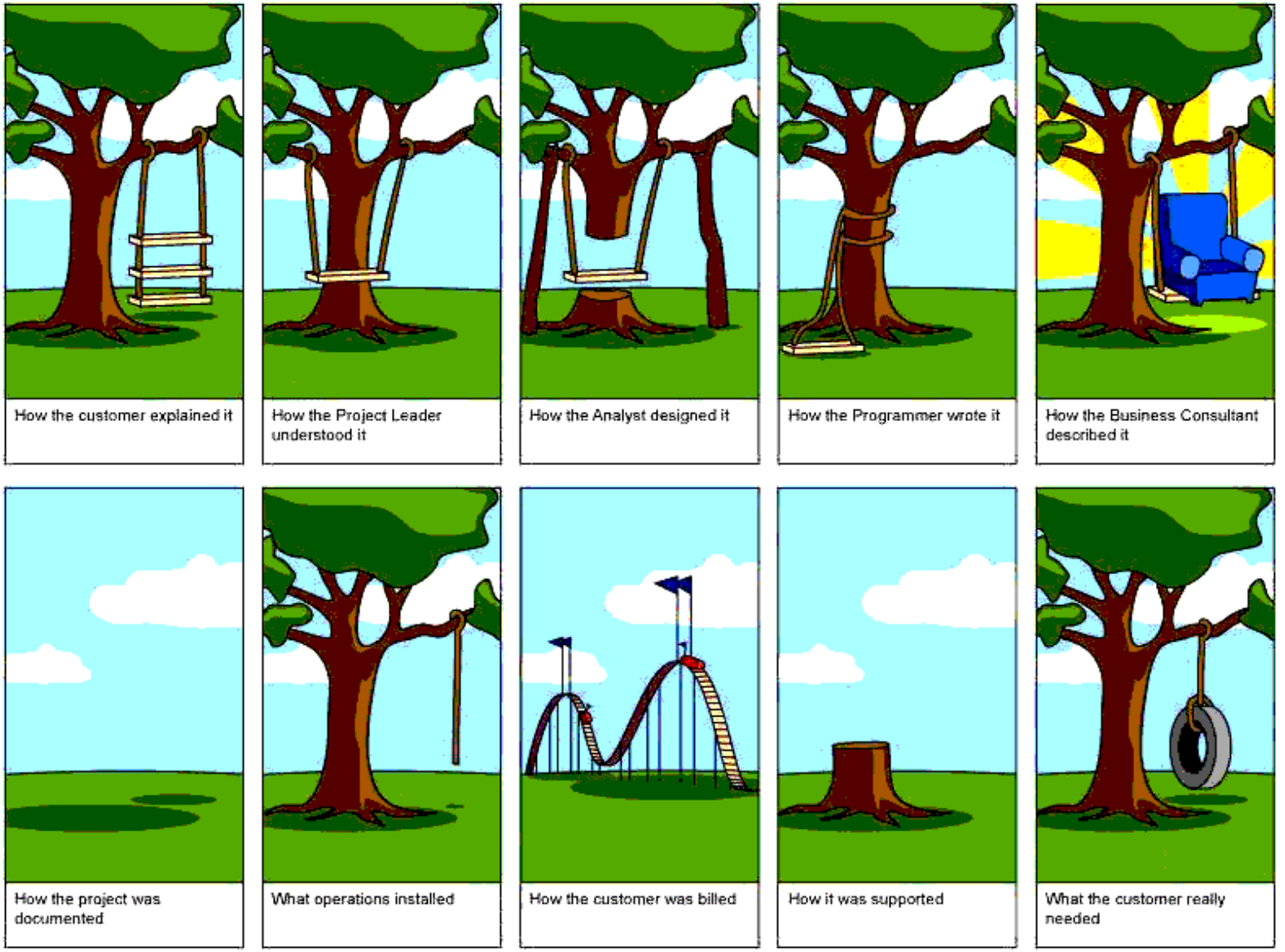
The same applies to writing a program too!

Understand the Problem

- Most important step!
  - What information do we know?
  - What do we want to happen?
  - How can one lead to the other?

- In programming
  - this involves talking to customers
  - what do they want?

Otherwise…


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Understanding a Programming Problem

- For the programming exercises
  - do you understand what you are required to do
  - do you understand the programming constructs you will need to use
    - (if not make sure you understand them before you start)
- do not start to write a program until you have understood the problem

Different Solutions

- I want to eat Pizza

  … I have lots of options

  - Go to a Pizza Restaurant
  - Phone a Pizza delivery place
  - Cook a frozen one
  - Make one myself

Same Plans - Different Details

- Pizza Restaurant
  - Pizza Express v Pizza Hut
  - Leicester Square v local
- Frozen
  - buy one with toppings
  - add your own toppings
- Make your own
  - different recipes for base

Different Solutions

- There are many different ways to achieve a task
- Trade-off between advantages and disadvantages of each
- What is best depends on the situation
- Given a plan
  - still different ways to do it
- Programming is about devising plans
  - the above all apply

Algorithms

A plan for solving a problem by following a sequence of rules is called an algorithm

- An algorithm should be (at least)
  - **complete (i.e. cover all the parts)**
  - **unambiguous (no doubt about what it does)**
  - **deterministic (only 1 possible result)**
  - **finite (it should finish)**

Program Planning

1. Analyze: Define the problem.
2. Design: Plan the solution to the problem.
3. Code: Translate the algorithm into a programming language.
4. Test and correct: Locate and remove any errors in the program.
5. Complete the documentation: Organize all the material that describes the program.

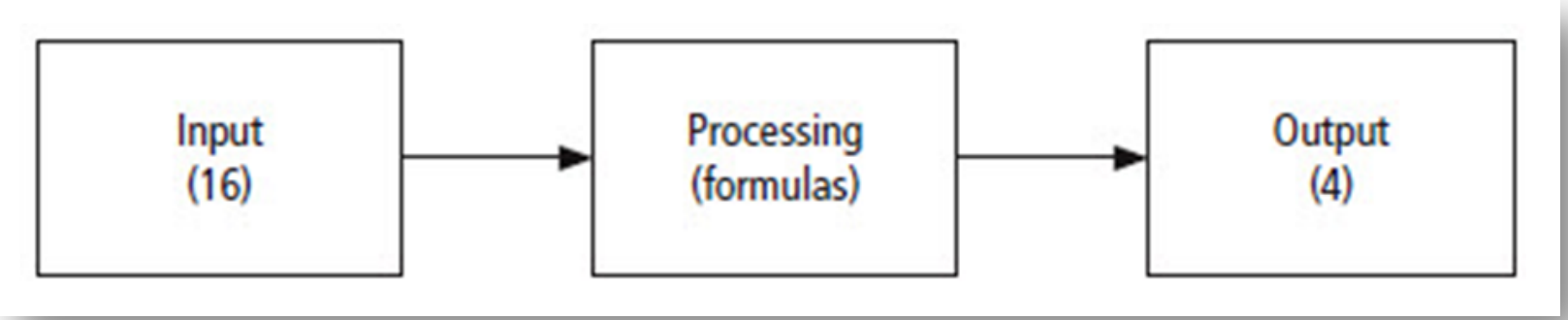
Programming Tools

- Algorithms
- Flowcharts
- Pseudocode
- Hierarchy charts

Algorithm Development

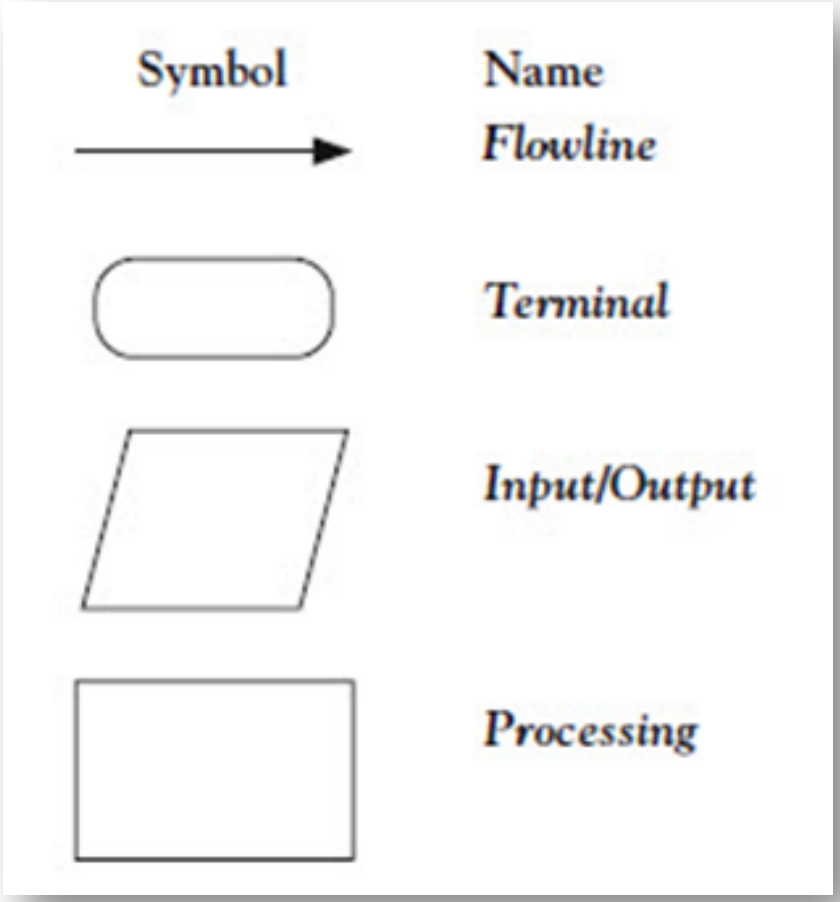
- Algorithm to determine number of stamps for a letter
- Rule of thumb: 1 stamp for every 5 sheets of paper
  1. Request sheets of paper
  2. Divide by 5
  3. Round quotient up to next whole number
  4. Reply with number of stamps

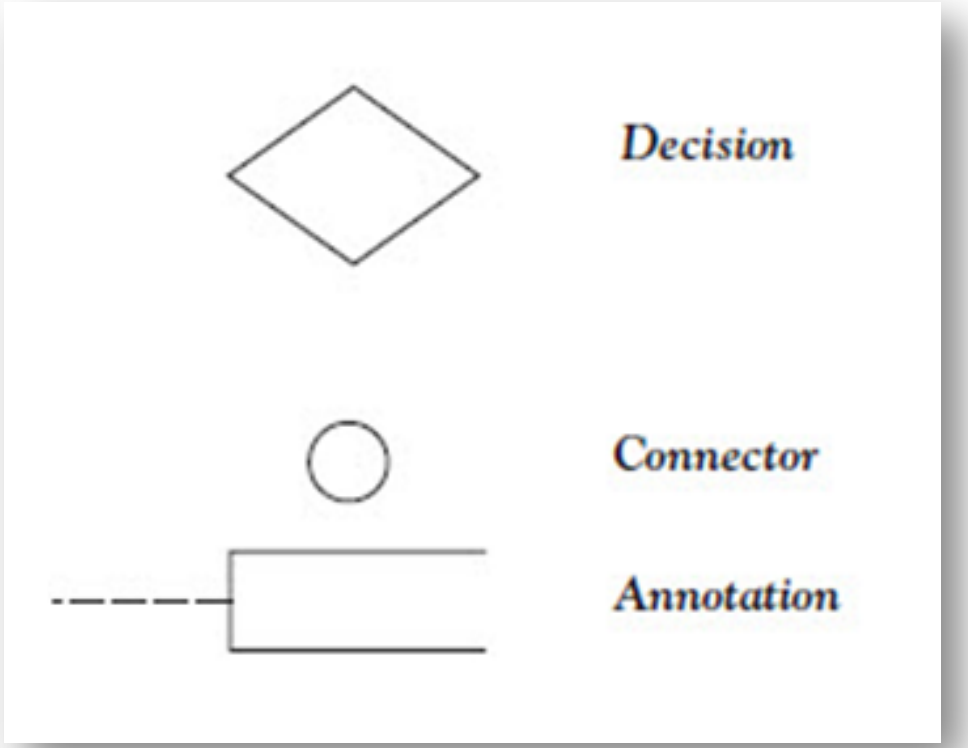
Problem Solving for Stamps



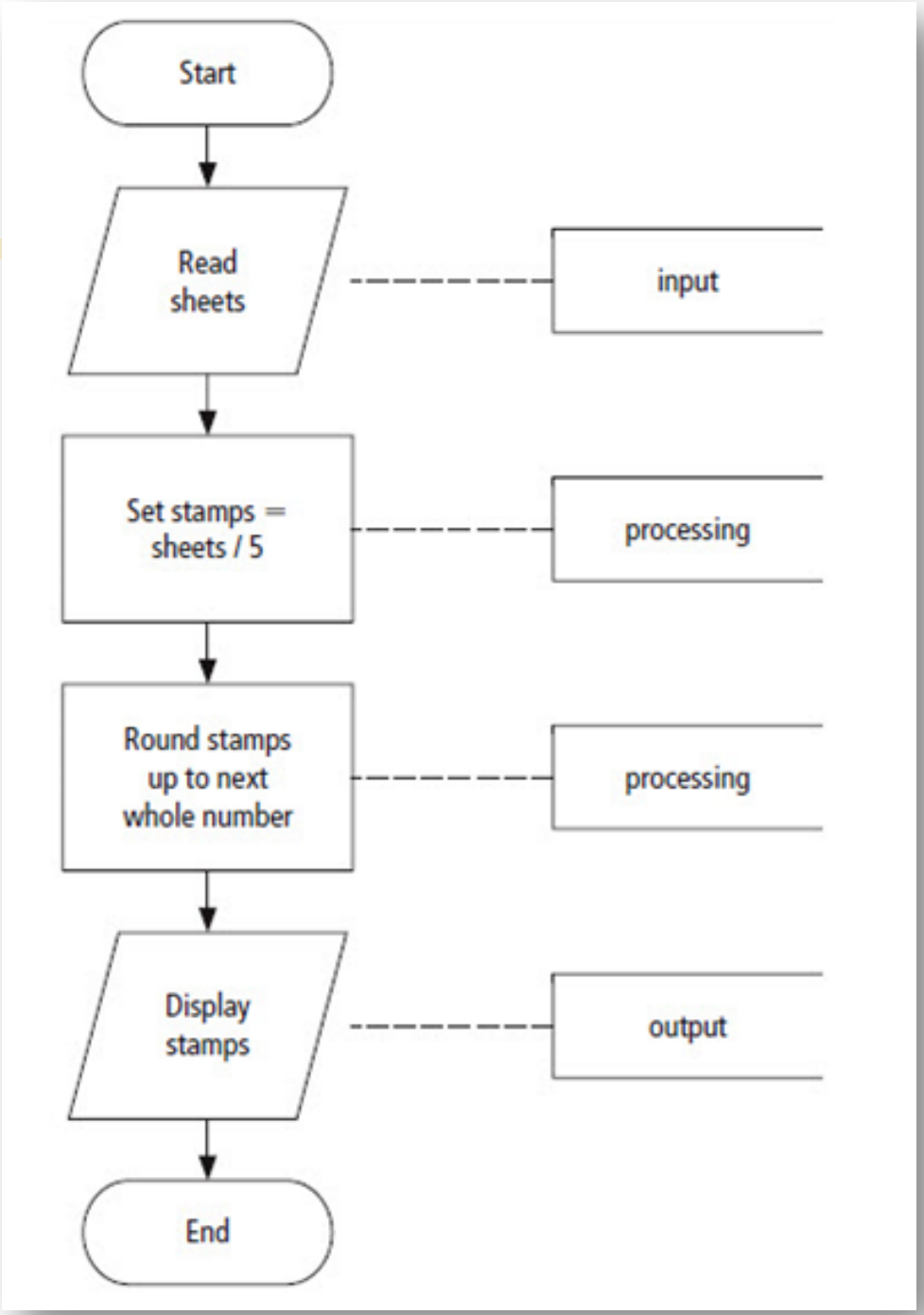
The problem solving process
for the stamp problem.

Flowcharts


Symbol
Name
Flowline
Terminal
Input/Output
Processing


Decision
Connector
Annotation

Example Flowchart



Flowchart for the postage-stamp problem.

Pseudocode

- Abbreviated plain English version of actual computer code
- Symbols used in flowcharts replaced by English-like statements
- Allows programmer to focus on steps required to solve problem

Example Pseudocode

Program: Determine the proper number of stamps for a letter.

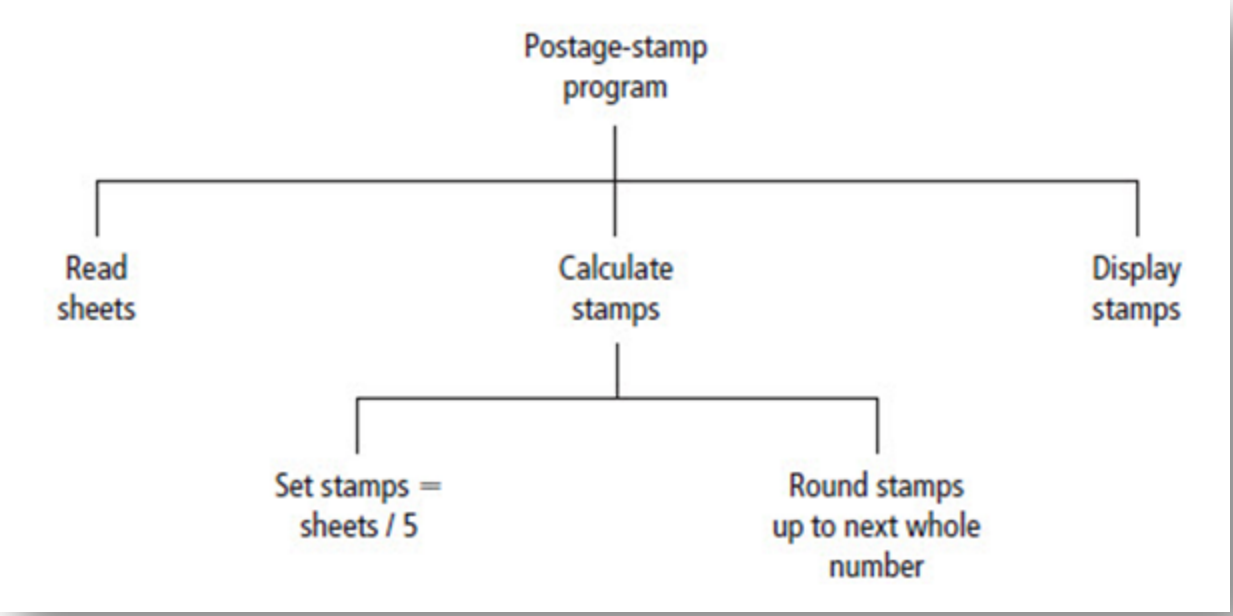
Read Sheets	(*input*)
Set the number of stamps to Sheets / 5	(*processing*)
Round the number of stamps up to the next whole number	(*processing*)
Display the number of stamps	(*output*)

Pseudocode for the postage stamp problem.

Hierarchy Chart

- Shows the overall program structure
- Depict organization of program, omit specific processing logic
- Describe what each part, or module, of the program does
- Each module subdivided into a succession of submodules

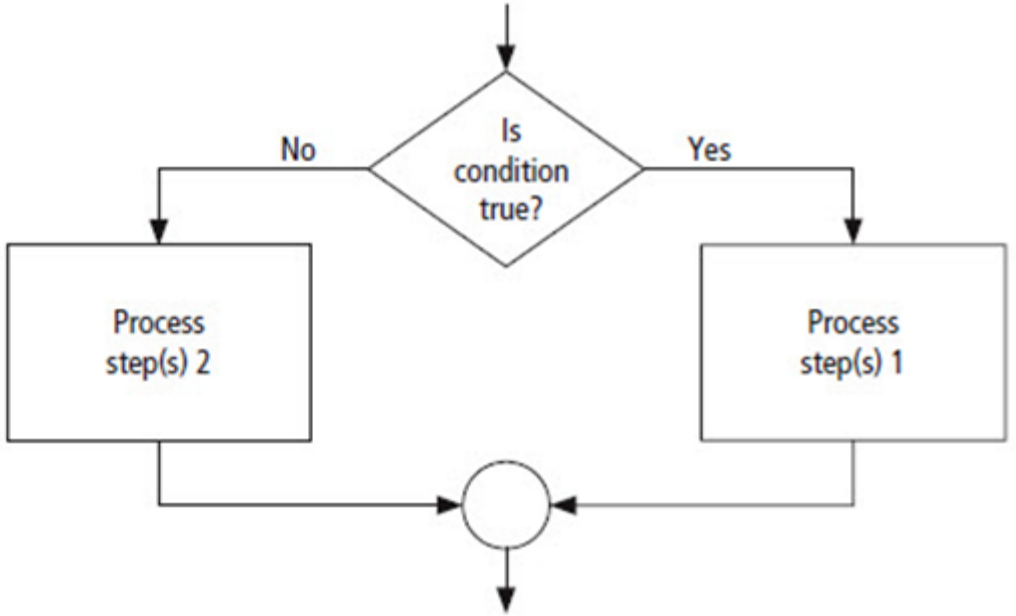
Example Hierarchy Chart



Hierarchy chart for the postage-stamp problem.

Decision Structure

```
if condition is true
   Process step(s) 1
else
   Process step(s) 2
```



Pseudocode and flowchart for a decision structure.

Direction of Numbered NYC Streets Algorithm

- **Problem:** Given street number of one-way street in New York City, decide direction of street, either eastbound or westbound.
- **Discussion:** There is a simple rule to tell the direction of a one-way street in New York City: Even-numbered streets run eastbound.

Direction of Numbered NYC Streets Algorithm

- **Input:** Street number.
- **Processing:** Decide if the street number is divisible by 2.
- **Output:** "Eastbound" or "Westbound".

Direction of Streets Algorithm

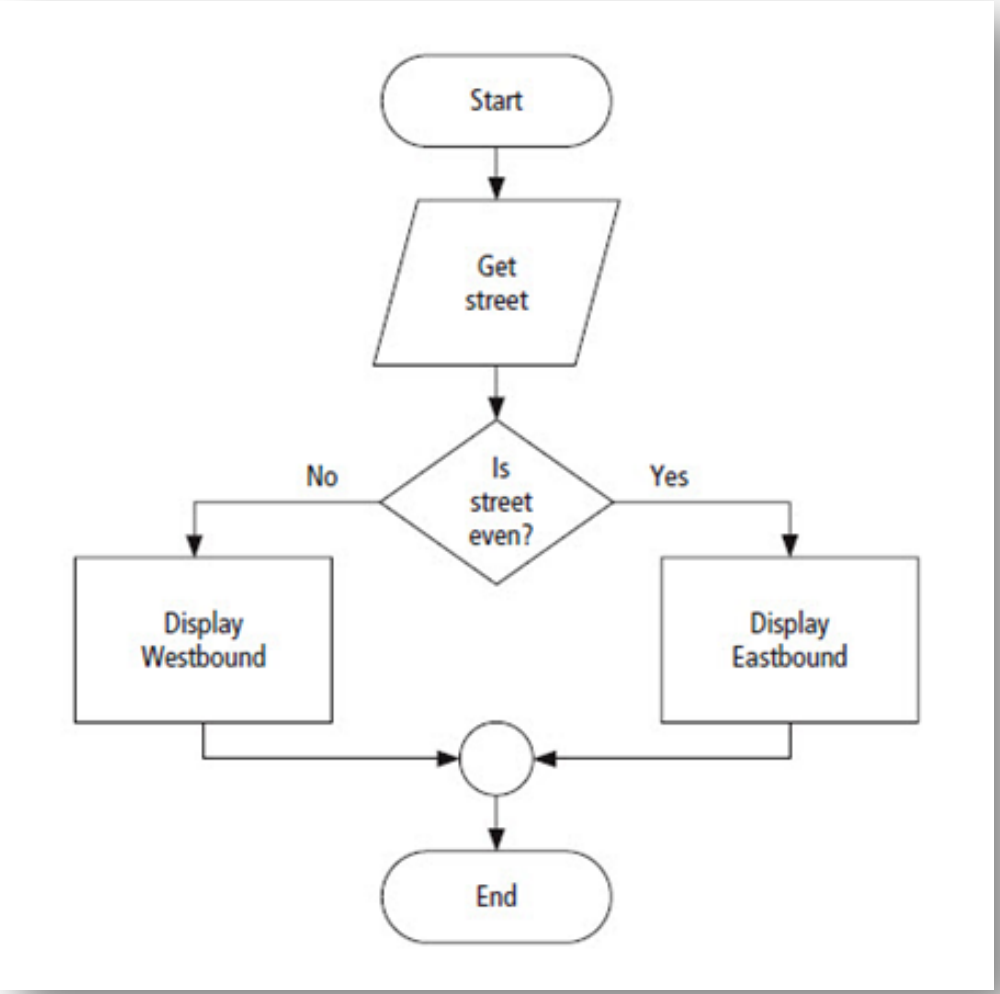


FIGURE 1.8
Flowchart for the numbered New York City streets problem.

Direction of Streets Algorithm

Program: Determine the direction of a numbered NYC street.

```
Get street
if street is even
   Display Eastbound
else
   Display Westbound
```

FIGURE 1.9 Pseudocode for the numbered New York City streets problem.

Direction of Streets Algorithm

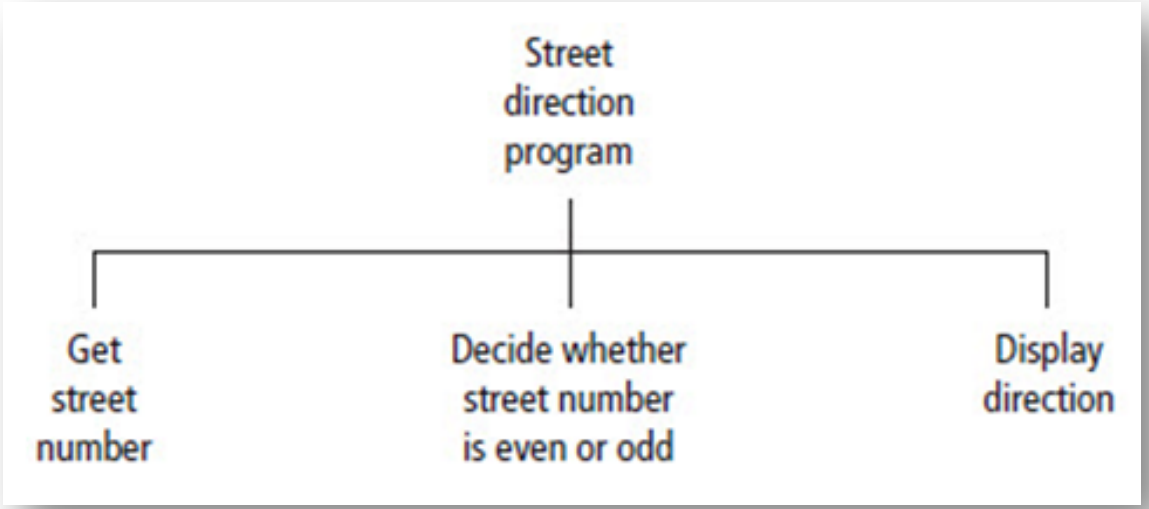


FIGURE 1.10 Hierarchy chart for the numbered New York City streets problem.

Repetition Structure

- A programming structure that executes instructions many times
  - Repetition structure
  - Looping structure
- Need a test (or condition) to tell when the loop should end
  - Check condition before each pass through loop

Direction of Streets Algorithm

```
while condition is true
    Process step(s)
```

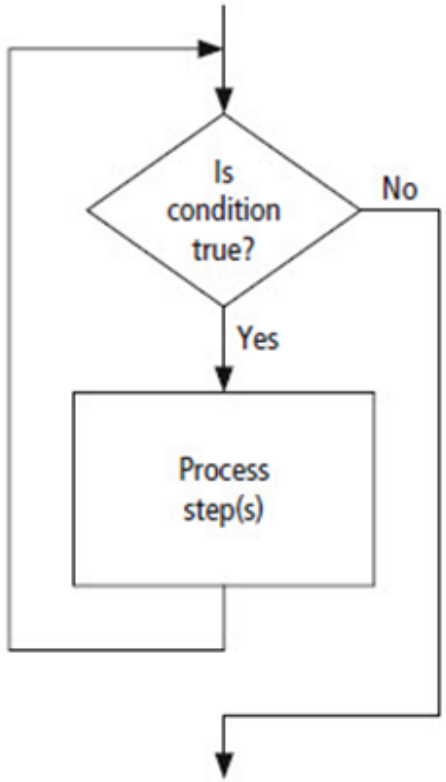


FIGURE 1.11 Pseudocode and flowchart for a loop.

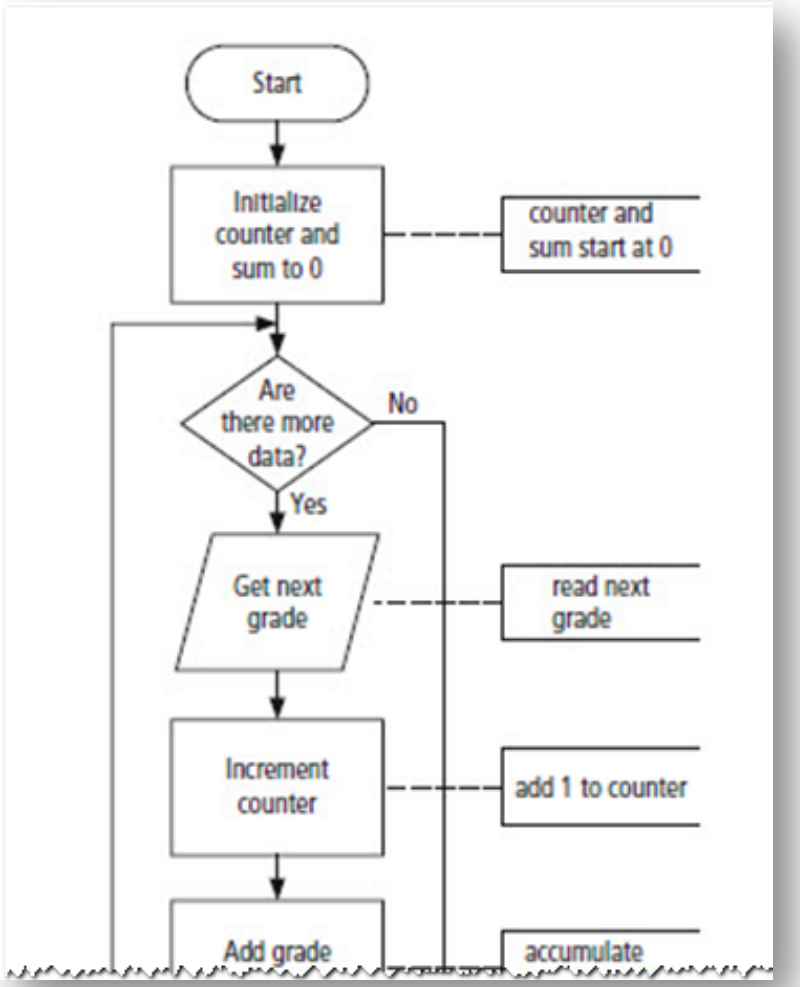
Class Average Algorithm

- **Problem:** Calculate and report the average grade for a class.
- **Discussion:** Average grade equals sum of all grades divided by number of students.
  - Need loop to read and then add (accumulate) grades for each student in class.
  - Inside the loop, we also need to total (count) number of students in class.

Class Average Algorithm

- **Input:** Student grades.
- **Processing:** Find the sum of the grades; count the number of students; calculate average grade = sum of grades / number of students.
- **Output:** Average grade

Class Average Algorithm



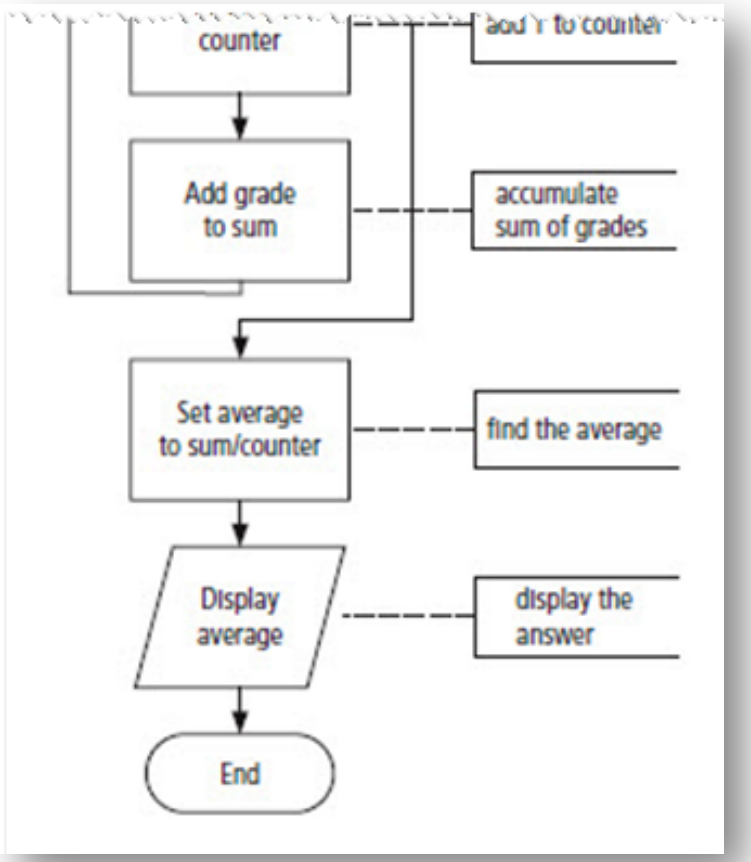


FIGURE 1.12 Flowchart for the class average problem.

Class Average Algorithm

Program: Calculate and report the average grade of a class.

```
Initialize Counter and Sum to 0
while there are more data
    Get the next Grade
    Increment the Counter
    Add the Grade to the Sum
    Compute Average = Sum/Counter
    Display Average
```

FIGURE 1.13 Pseudocode for the class average problem.

Class Average Algorithm

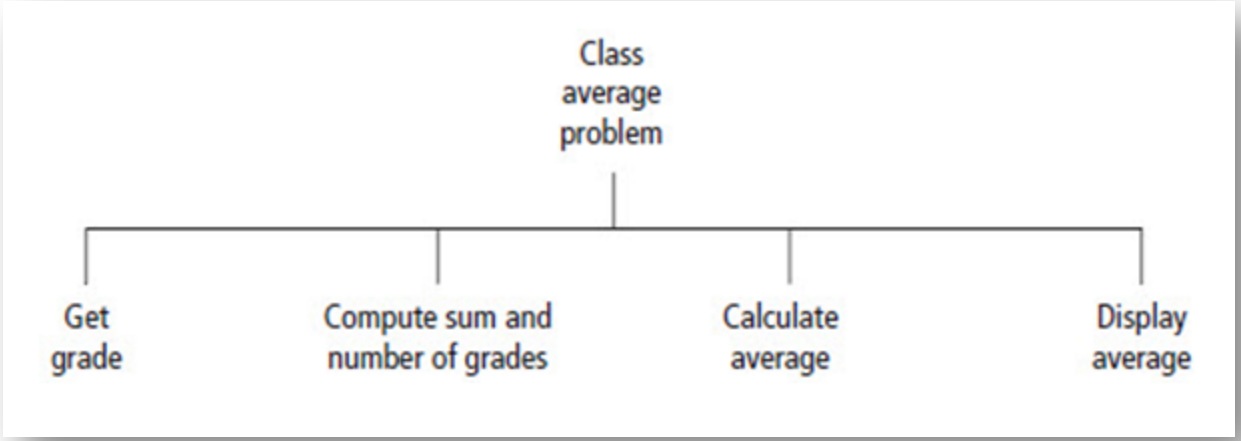


FIGURE 1.14 Hierarchy chart for the class average problem.

Exercise

- Look at the Noughts and Crosses Pseudocoding exercise at myCourses.
- Form groups of three and try to write a set of instructions.
- Make sure you share your work at the end.

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

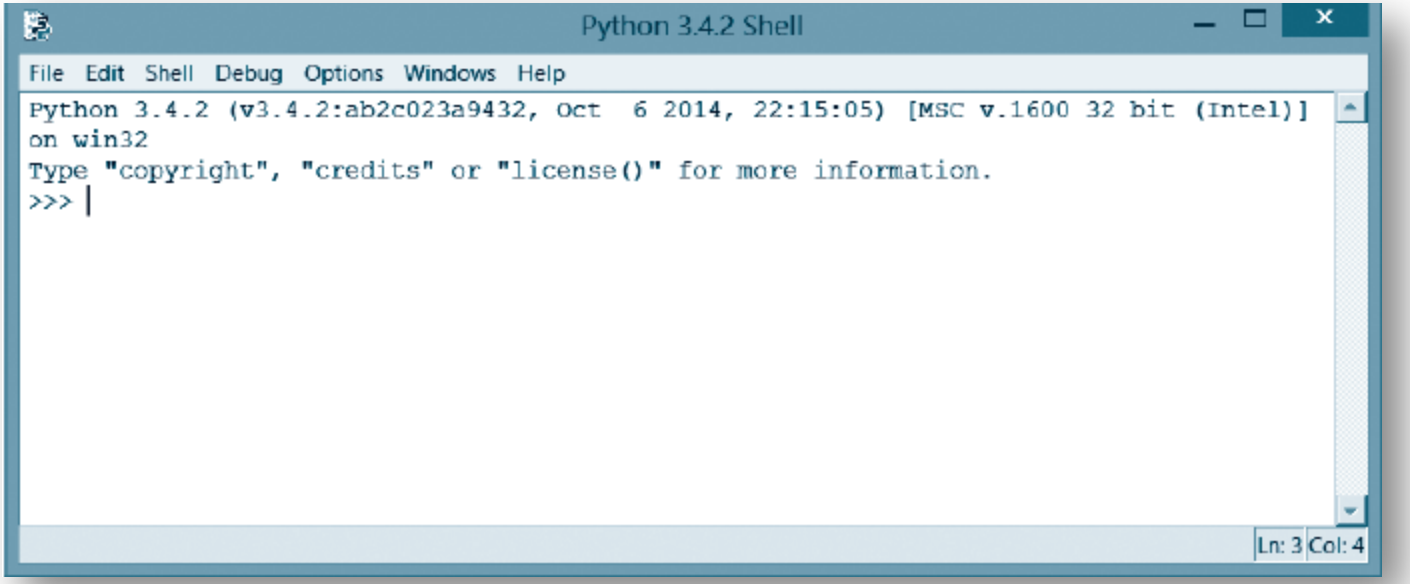


FIGURE 1.16 The Python shell.

Starting IDLE

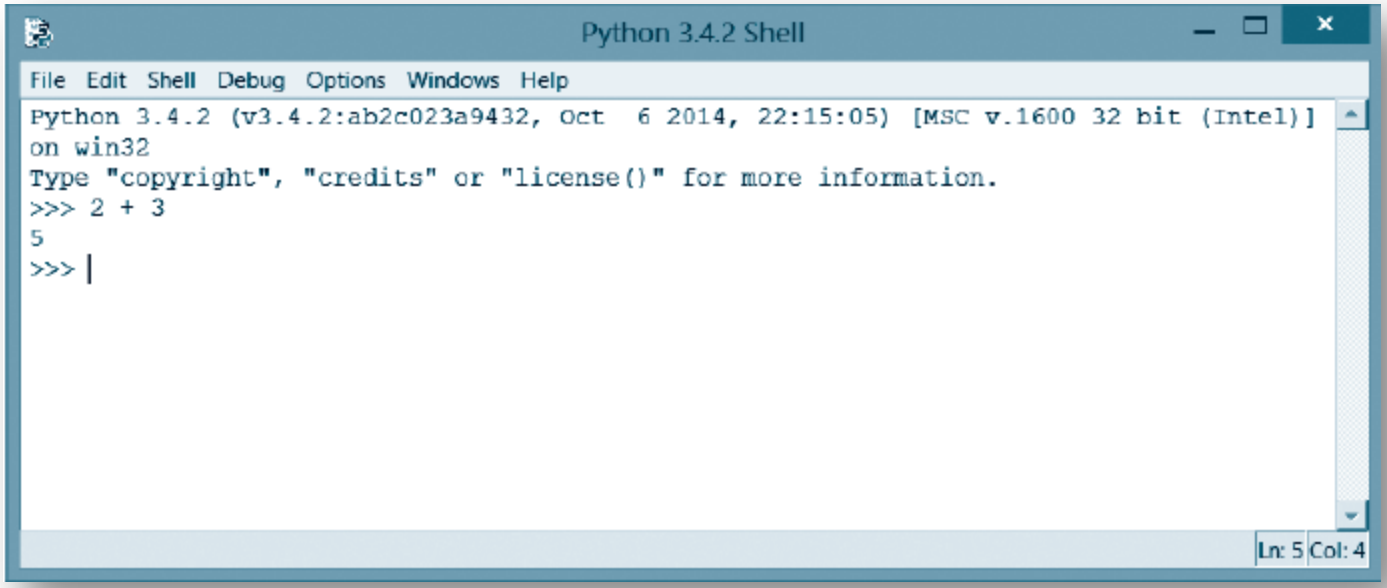


FIGURE 1.17 The Python shell after the expression 2 + 3 has been evaluated.

Starting IDLE

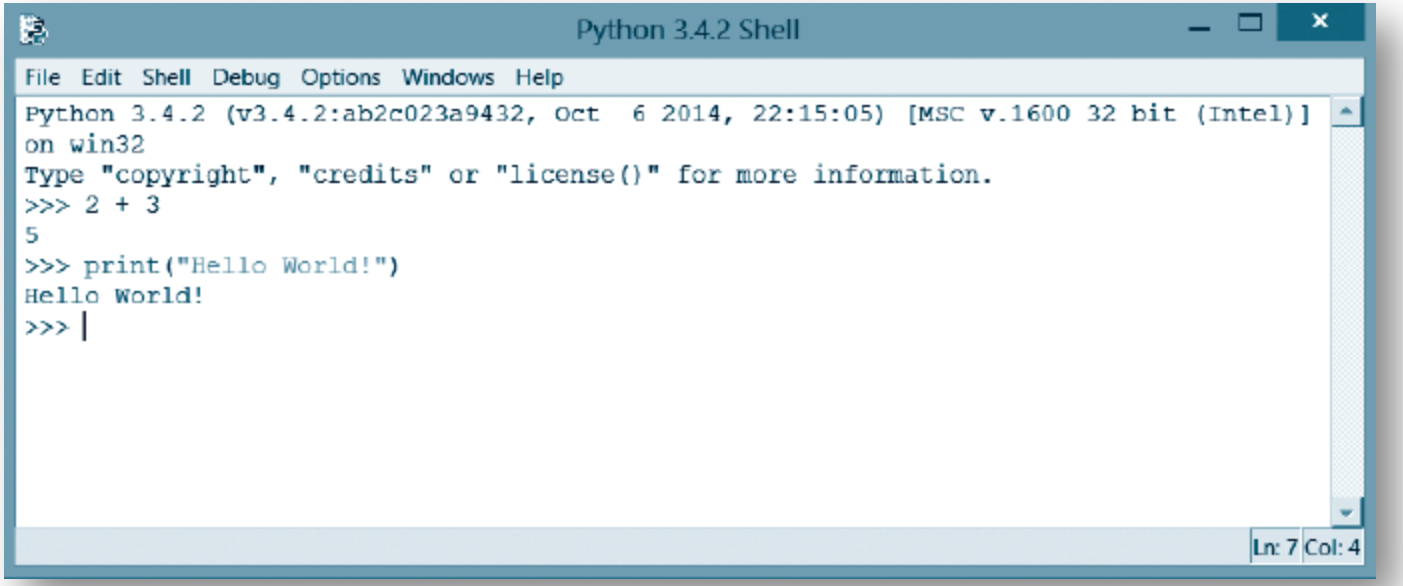


FIGURE 1.18 The Python shell after the statement print("Hello World!") has been executed.

A Python Code Editor Walkthrough

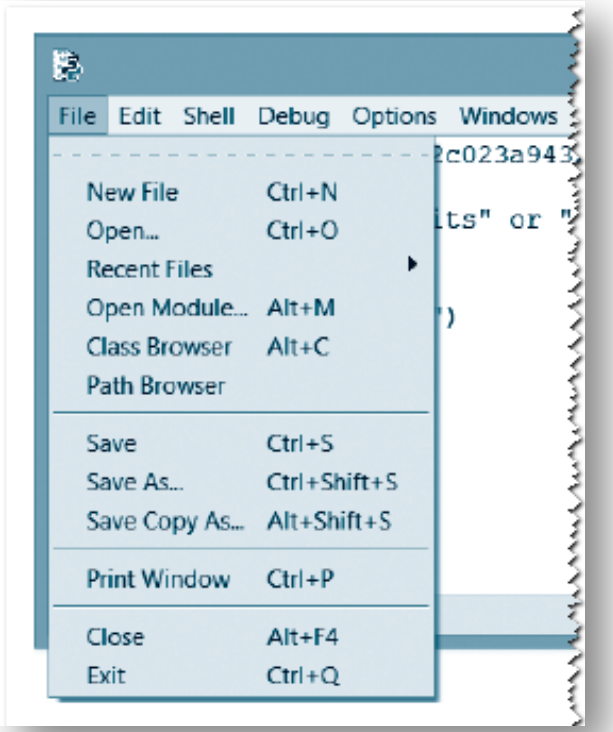


FIGURE 1.19 The File drop-down list.

A Python Code Editor Walkthrough

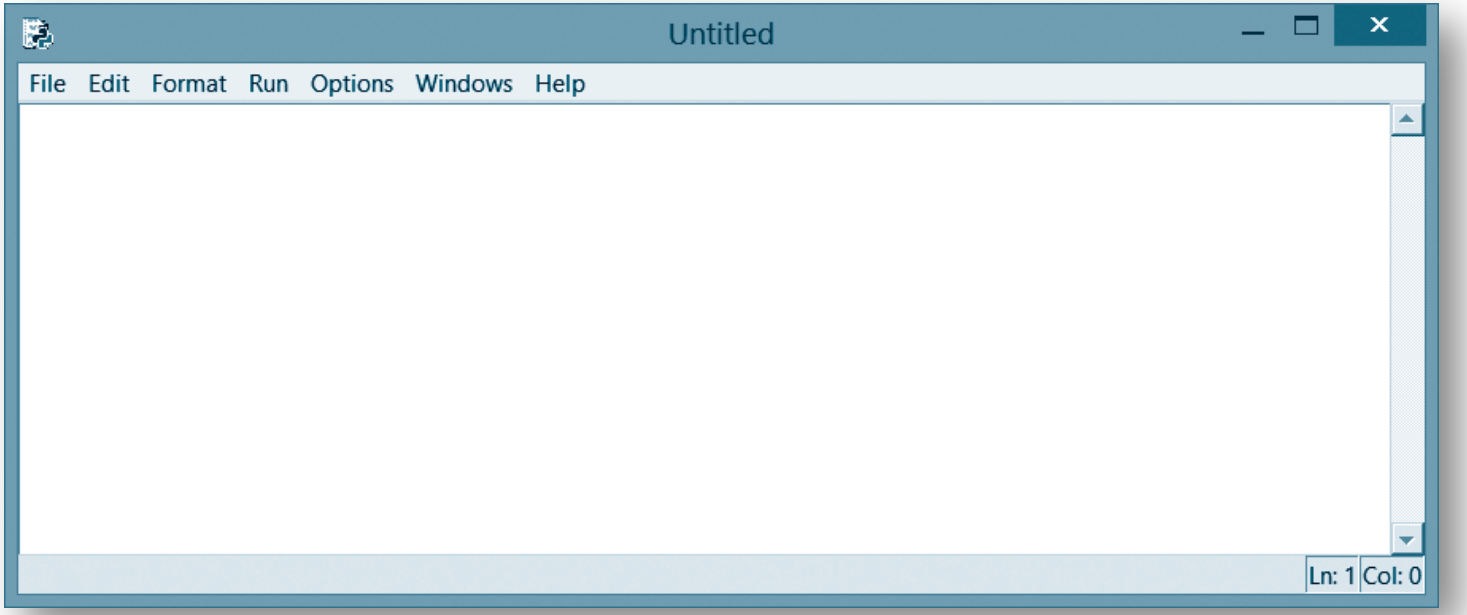


FIGURE 1.20 The code editor window generated after New Window is clicked on.

A Python Code Editor Walkthrough

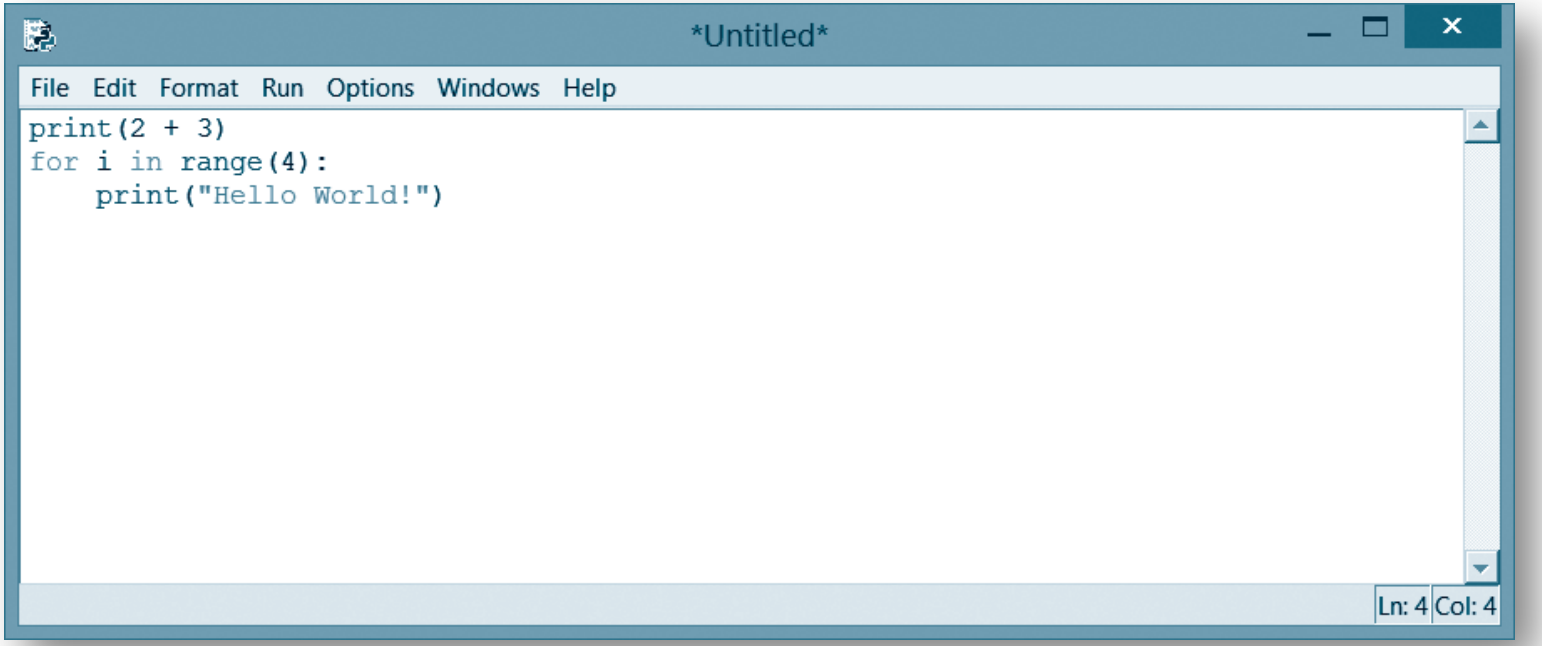


FIGURE 1.21 The code editor window containing a three-line Python program.

A Python Code Editor Walkthrough

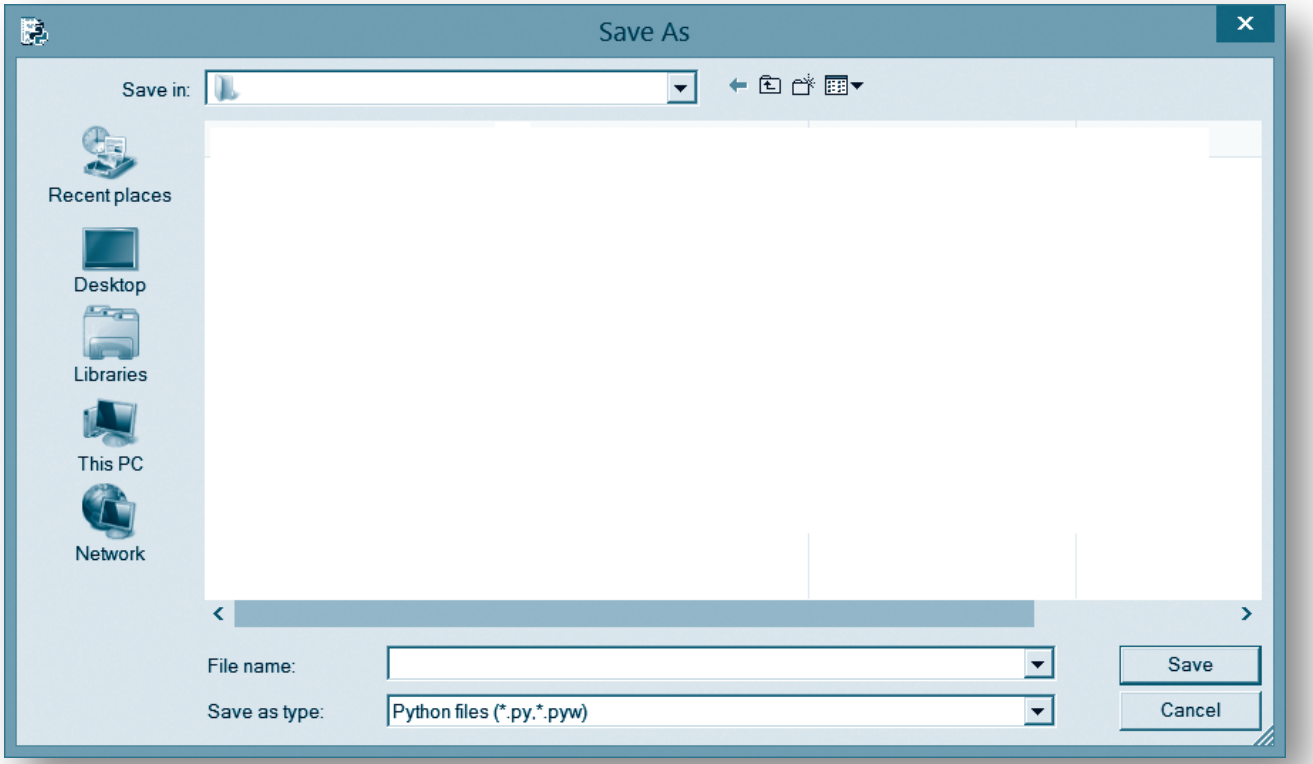


FIGURE 1.22 A Save As dialog box.

A Python Code Editor Walkthrough

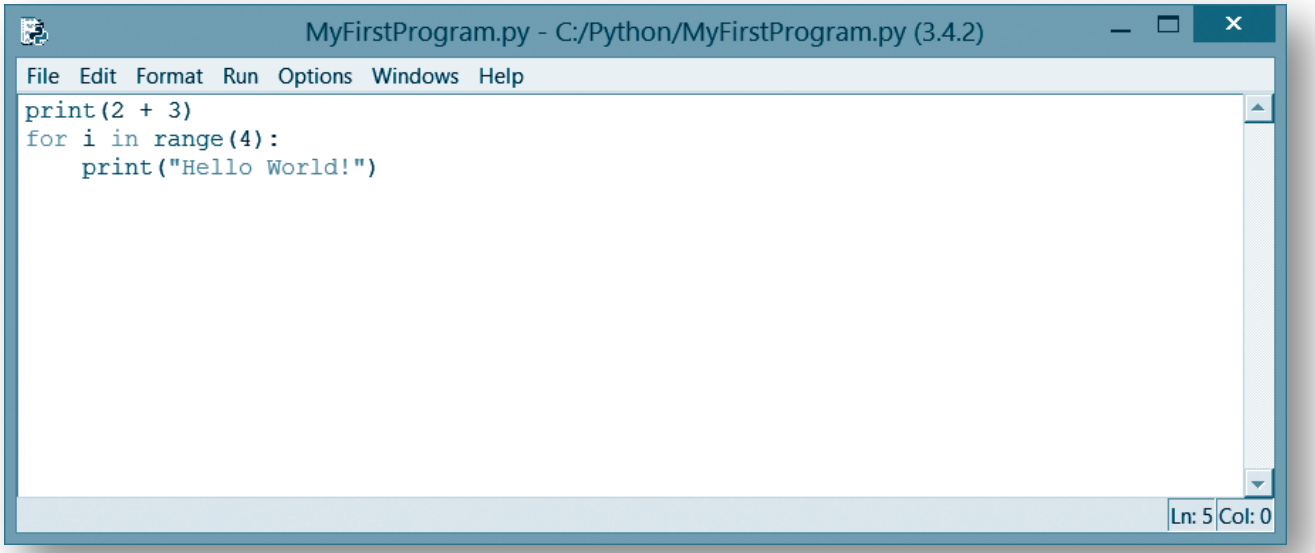


FIGURE 1.23 The code editor window containing a three-line Python program.

A Python Code Editor Walkthrough

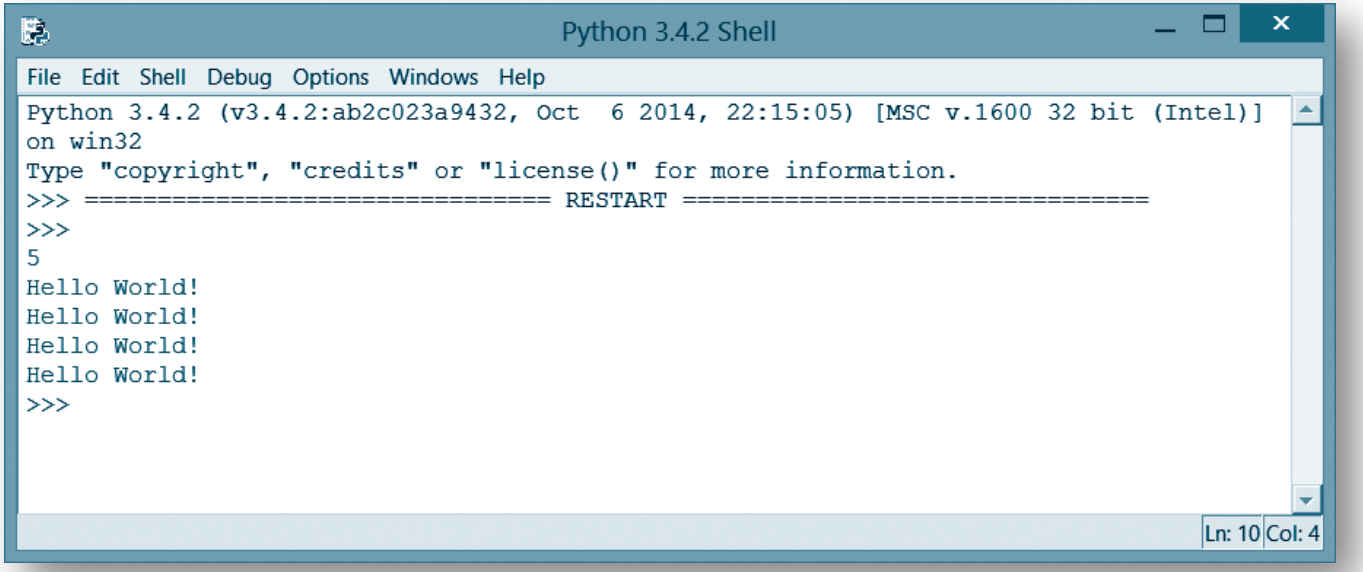


FIGURE 1.24 Press the F5 key to execute. The outcome of the Python program in Fig. 1.22.

A Python Code Editor Walkthrough

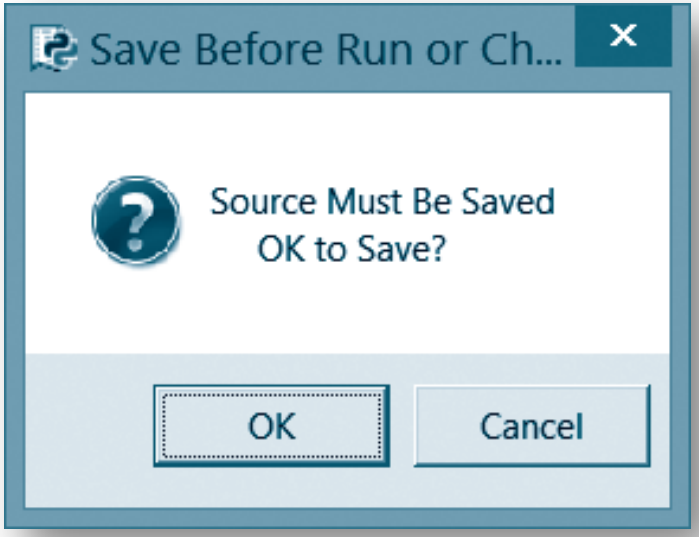


FIGURE 1.25 A Save message box.

An Open-a-Program Walkthrough

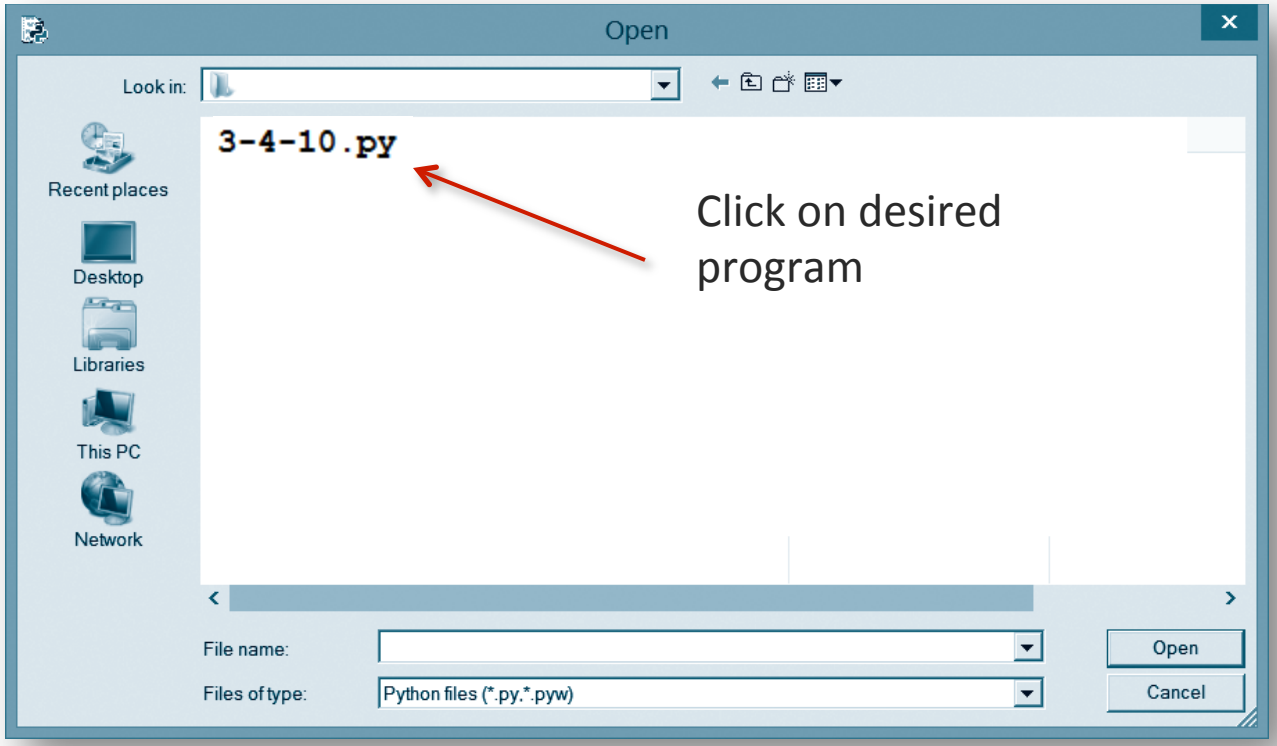


FIGURE 1.26 An Open dialog box.

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

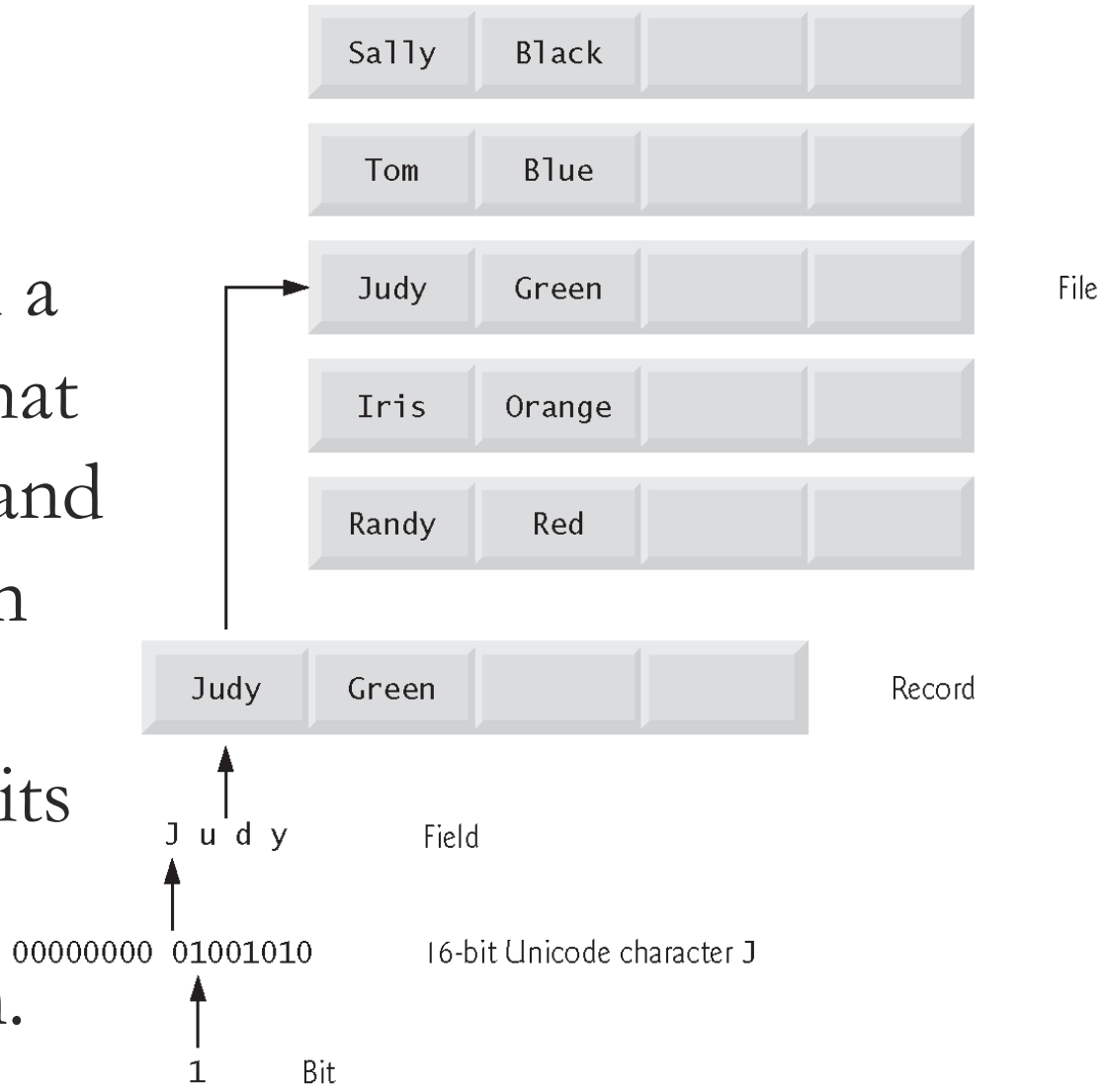
Logical unit	Description
Arithmetic and logic unit (ALU)	This "manufacturing" section performs *calculations*, such as addition, subtraction, multiplication and division. It also contains the *decision* mechanisms that allow the computer, for example, to compare two items from the memory unit to determine 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.