Python - Week 1

Mohammad Shokoohi-Yekta

An Introduction to Computers and Problem Solving

- 1.1 An Introduction to Computers
- 1.2 Program Development Cycle
- 1.3 Programming Tools
- 1.4 Starting Python

Communicating with the Computer

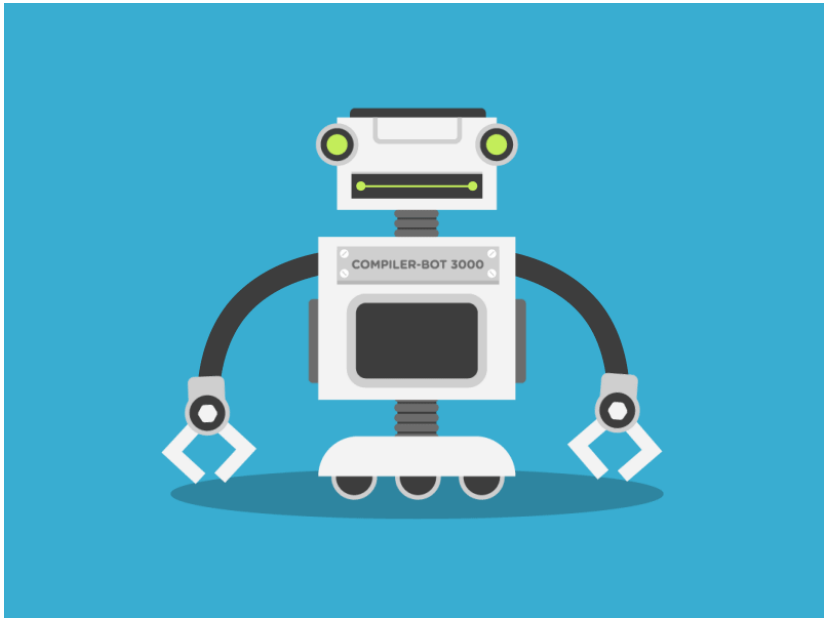
- Machine language – low level, hard for humans to understand
- Python – high level, understood by humans, consists of instructions such as Click, If, and Do

Popular High-Level Languages

- COBOL (COmmon Business Oriented Language)
- FORTRAN (FORmula TRANslation)
- BASIC (Beginner All-purpose Symbolic Instructional Code)
- Pascal (named for Blaise Pascal)
- Ada (named for Ada Lovelace)
- C (whose developer designed B first)
- Visual Basic (Basic-like visual language developed by Microsoft)
- Delphi (Pascal-like visual language developed by Borland)
- C++ (an object-oriented language, based on C)
- C# (a Python-like language developed by Microsoft)
- Python (We use it in the book)

Compiler

- A compiler translates a high-level language into machine language.
- The Python compiler points out certain types of errors during the translation process.



Programming and Complicated Tasks

- Tasks are broken down into instructions that can be expressed by a programming language
- A *program* is a sequence of instructions
- Programs can be only a few instructions or millions of lines of instructions

All Programs Have in Common:

- Take data and manipulate it to produce a result
- Input – Process – Output
  - Input – from files, the keyboard, or other input device
  - Output – usually to the monitor, a printer, or a file

Hardware and Software

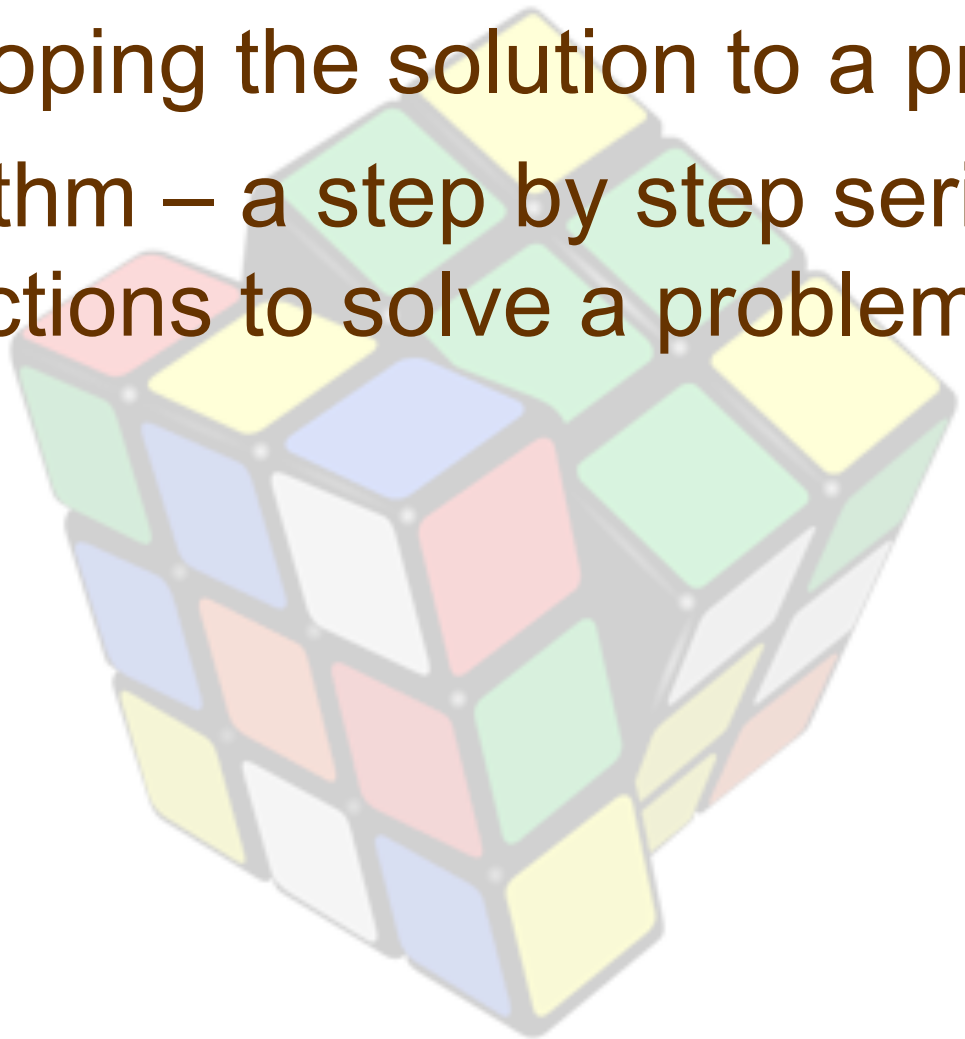
- Hardware – the physical components of the computer
  - Central processing unit
  - Disk drive
  - Monitor
- Software – The instructions that tell the computer what to do

Programmer and User

- Programmer – the person who solves the problem and writes the instructions for the computer
- User – any person who uses the program written by the programmer

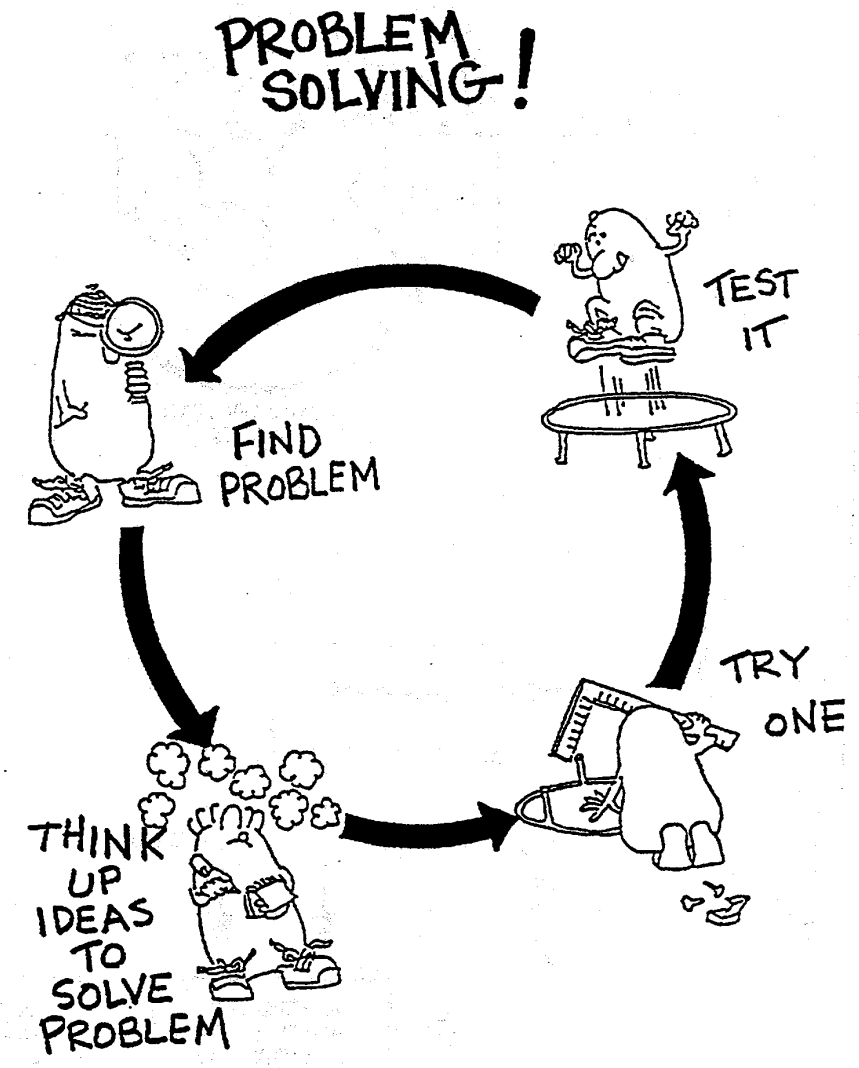
Problem Solving

- Developing the solution to a problem
- Algorithm – a step by step series of instructions to solve a problem



1.2 Program Development Cycle

- Performing a Task on the Computer
- Program Planning



Terminology

A computer program may also be called:

- Project
- Application
- Solution

Program Development Cycle

- Software refers to a collection of instructions for the computer
- The computer only knows how to do what the programmer tells it to do
- Therefore, the programmer has to know how to solve problems

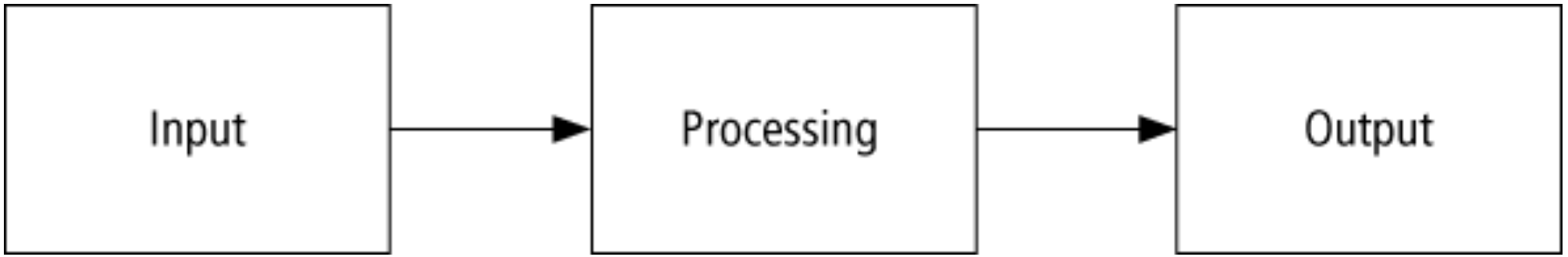
Performing a Task on the Computer

- Determine ***Output***
- Identify ***Input***
- Determine ***process*** necessary to turn given ***Input*** into desired ***Output***

Problem-Solving: Approach Like Algebra Problem

- How fast is a car traveling if it goes 50 miles in 2 hours?
- ***Output***: a number giving the speed in miles per hour
- ***Input***: the distance and time the car has traveled
- ***Process***: speed = distance / time

Pictorial representation of the Problem Solving Process



Program Planning

- A recipe is a good example of a plan
- Ingredients and amounts are determined by what you want to bake
- Ingredients are input
- The way you combine them is the processing
- What is baked is the output

Program Planning (continued)

- Always have a plan before trying to write a program
- The more complicated the problem, the more complex the plan must be
- Planning and testing before coding saves time

Program Development Cycle

1. Analyze: Define the problem.
2. Design: Plan the solution to the problem.
3. Choose the interface: Select the objects (text boxes, buttons, etc.).

Program Development Cycle (continued)

4. Code: Translate the algorithm into a programming language.
5. Test and debug: Locate and remove any errors in the program.
6. Complete the documentation: Organize all the materials that describe the program.

1.3 Programming Tools

- Flowcharts
- Pseudocode
- Hierarchy Chart
- Direction of Numbered NYC Streets Algorithm
- Class Average Algorithm

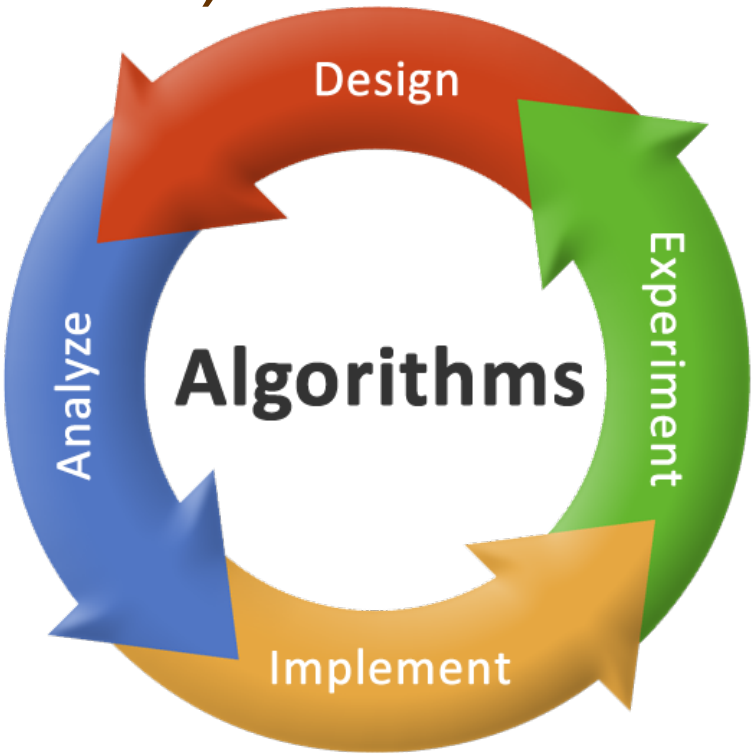
Programming Tools

Three tools are used to convert ***algorithms*** into computer programs:

- ***Flowchart*** - Graphically depicts the logical steps to carry out a task and shows how the steps relate to each other.
- ***Pseudocode*** - Uses English-like phrases with some Python terms to outline the program.
- ***Hierarchy chart*** - Shows how the different parts of a program relate to each other.

Algorithm

A step-by-step series of instructions for solving a problem (a recipe is an example of an algorithm).



Problem Solving Example

- How many stamps should you use when mailing a letter?
- One rule of thumb is to use one stamp for every five sheets of paper or fraction thereof.

Algorithm

1. Request the number of sheets of paper; call it Sheets. *(**input**)*
2. Divide Sheets by 5. *(**processing**)*
3. Round the quotient up to the next highest whole number; call it Stamps. *(**processing**)*
4. Reply with the number Stamps. *(**output**)*

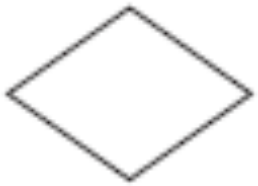
Flowchart

Graphically depicst the logical steps to carry out a task and show how the steps relate to each other.

Flowchart Symbols

Symbol	Name	Meaning
	*Flowline*	Used to connect symbols and indicate the flow of logic.
	*Terminal*	Used to represent the beginning (Start) or the end (End) of a task.
	*Input/Output*	Used for input and output operations, such as reading and displaying. The data to be read or displayed are described inside.
	*Processing*	Used for arithmetic and data-manipulation operations. The instructions are listed inside the symbol.

Flowchart Symbols (continued)

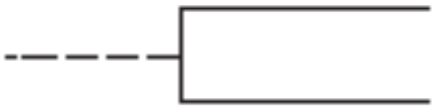


Decision

Used for any logic or comparison operations. Unlike the input/output and processing symbols, which have one entry and one exit flowline, the decision symbol has one entry and two exit paths. The path chosen depends on whether the answer to a question is "yes" or "no."

Connector

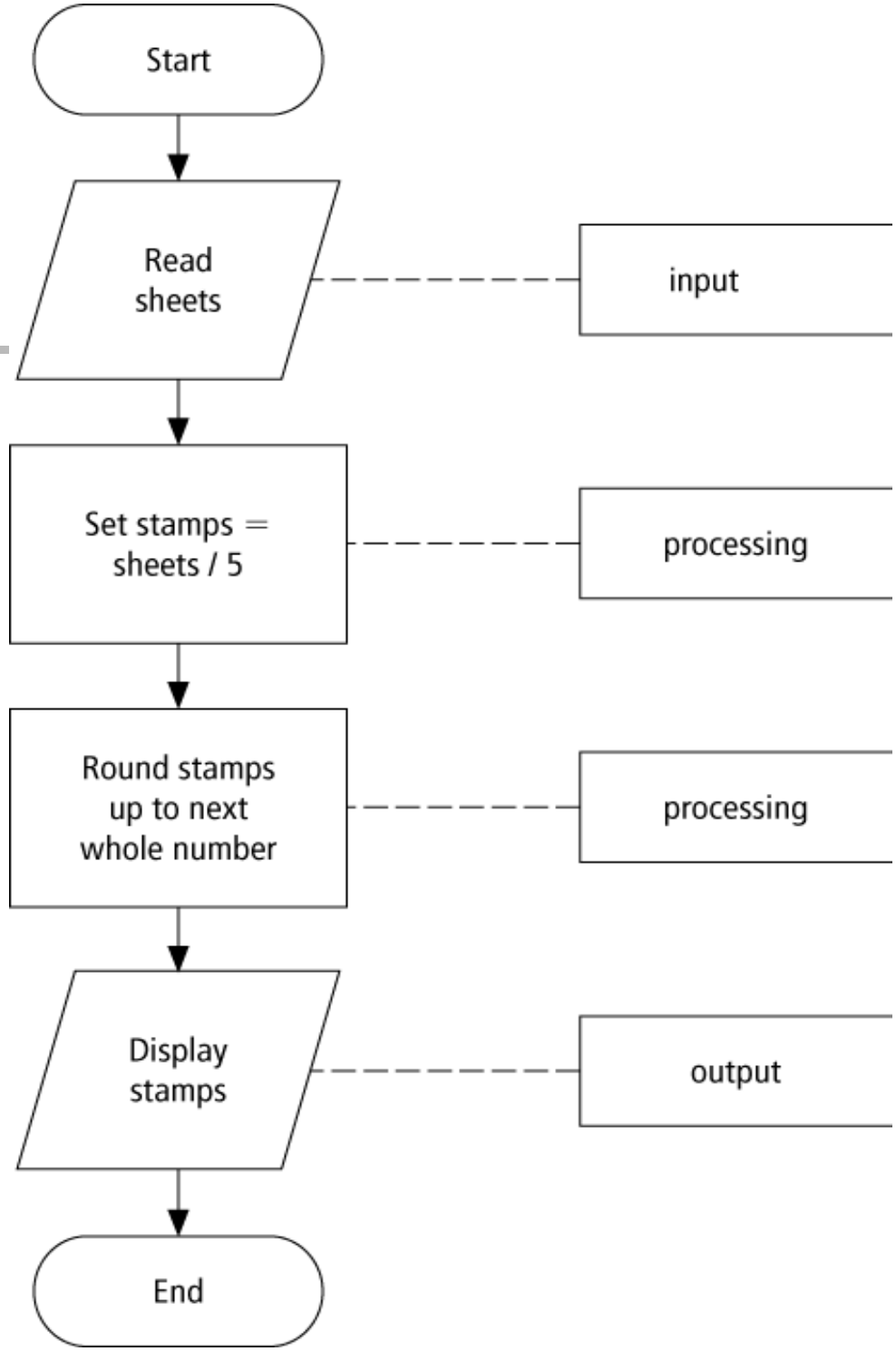
Used to join different flowlines.



Annotation

Used to provide additional information about another flowchart symbol.

Flowchart Example



Pseudocode

Uses English-like phrases with some Python terms to outline the task.

Pseudocode Example

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**)*

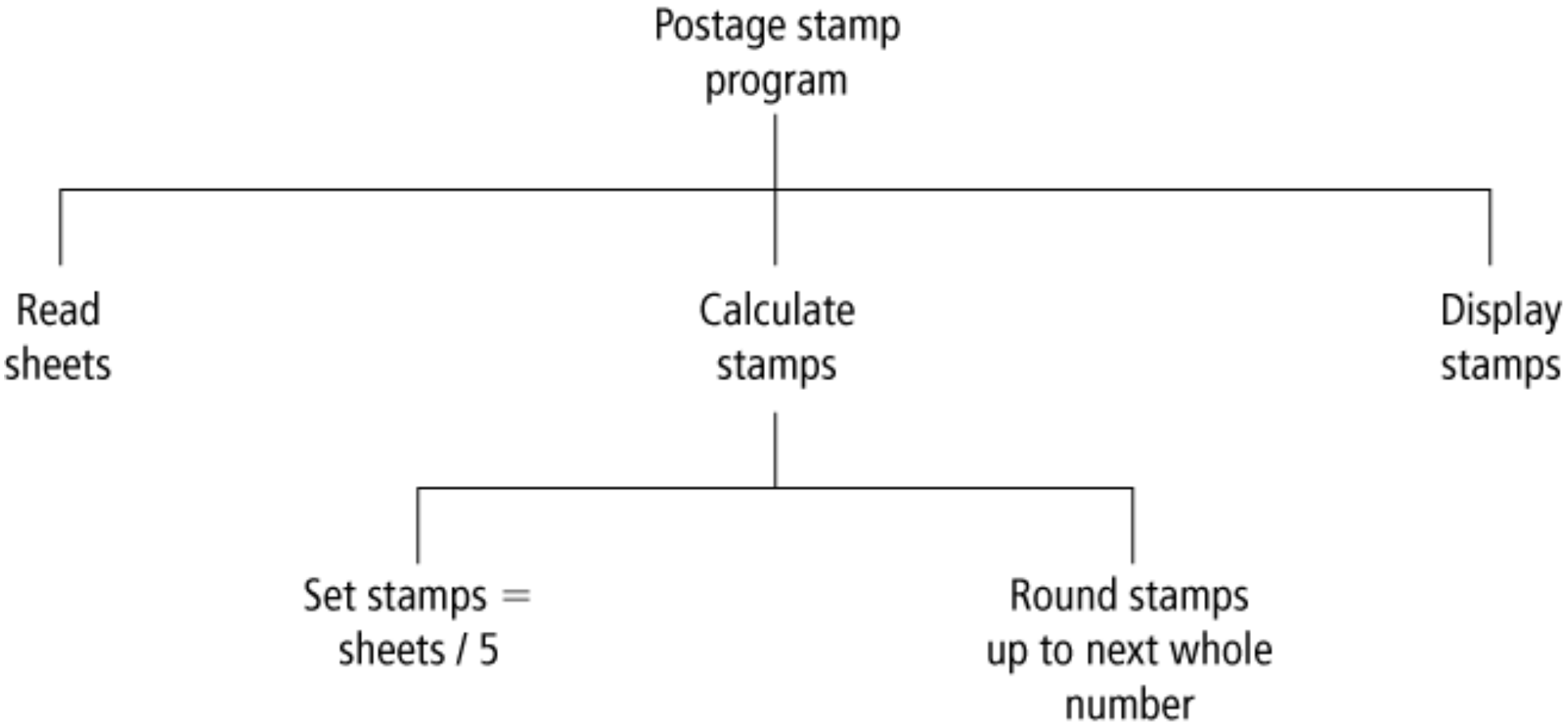
Hierarchy Chart

- Shows how the different parts of a program relate to each other

Hierarchy charts are also called

- structure charts
- HIPO (Hierarchy plus Input-Process-Output) charts
- top-down charts
- VTOC (Visual Table of Contents) charts

Hierarchy Charts Example



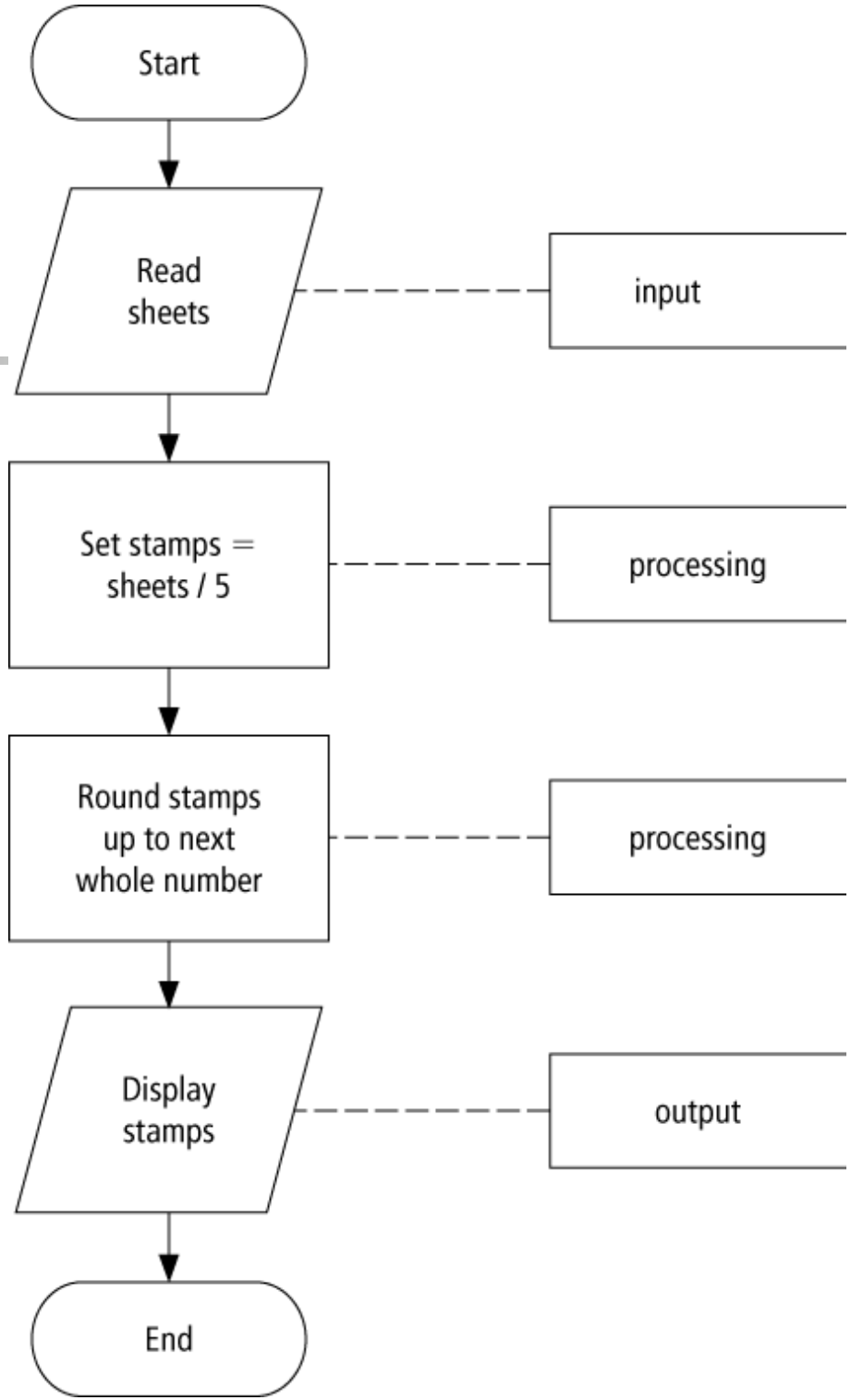
Divide-and-Conquer Method

- Used in problem solving – take a large problem and break it into smaller problems
- Solve the small problems first

Statement Structures

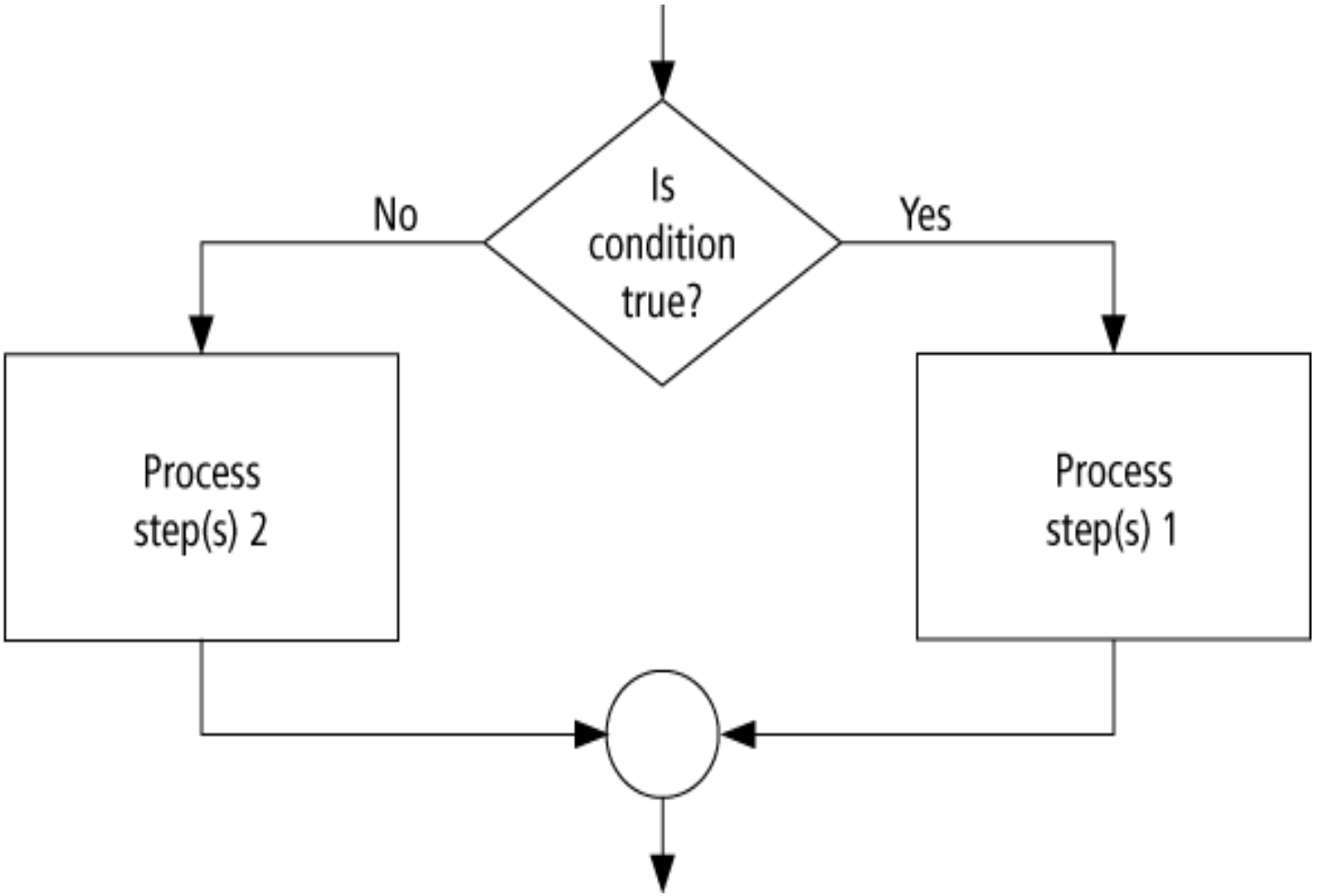
- Sequence – execute instructions from one line to the next without skipping over any lines
- Decision - if the answer to a question is “Yes” then one group of instructions is executed. If the answer is “No,” then another is executed
- Looping – a series of instructions are executed repeatedly

Sequence Flow Chart



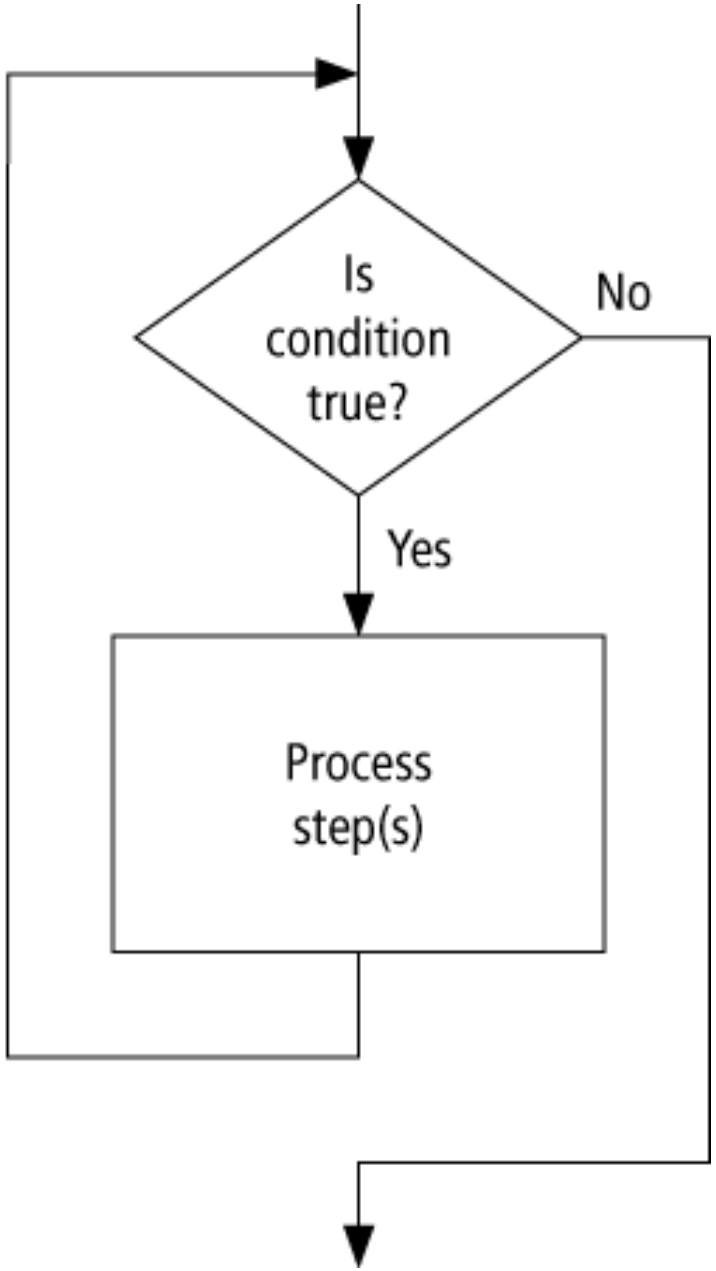
Decision Flow Chart

```
If condition is true Then
    Process step(s) 1
  Else
    Process step(s) 2
End If
```



Looping Flow Chart

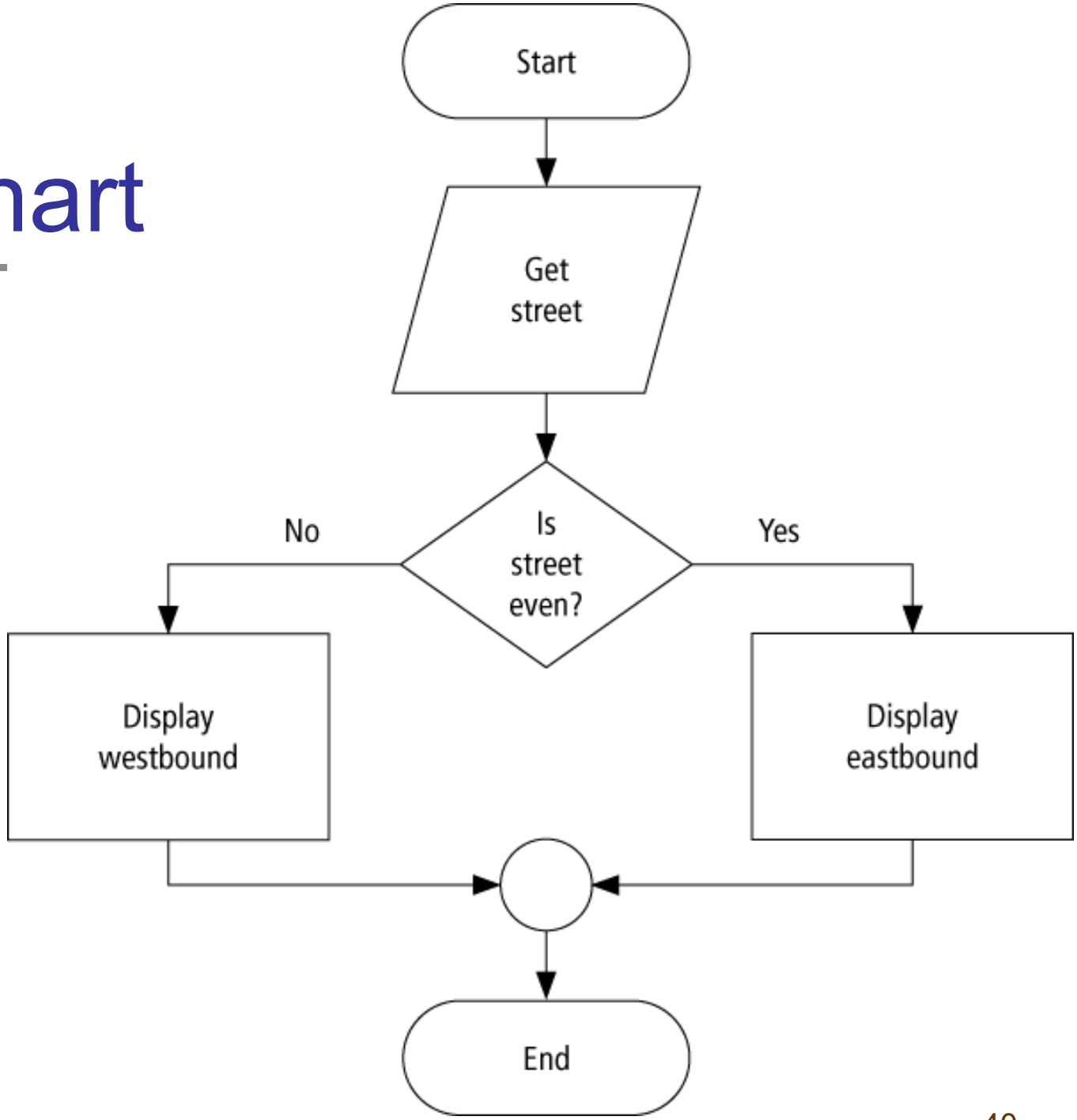
```
Do While condition is true
    Process step(s)
Loop
```



Direction of Numbered NYC Streets Algorithm

- ***Problem:*** Given a street number of a one-way street in New York City, decide the direction of the street, either eastbound or westbound
- ***Discussion:*** in New York City even numbered streets are Eastbound, odd numbered streets are Westbound

Flowchart


Start
Get street
Is street even?
No
Yes
Display westbound
Display eastbound
End

Pseudocode

Program: Determine the direction of a numbered NYC street

Get street

If street is even Then

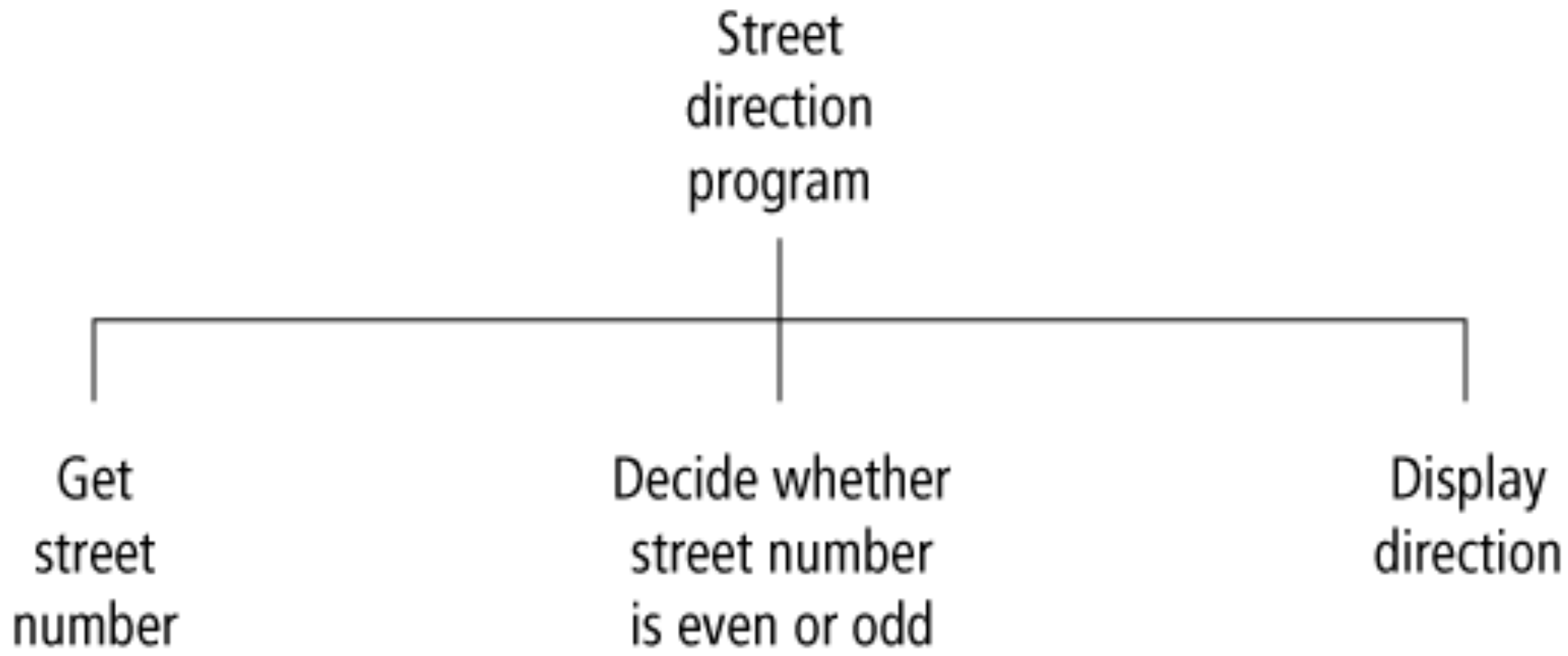
    Display Eastbound

Else

    Display Westbound

End If

Hierarchy Chart



Class Average Algorithm

Problem: Calculate and report the average grade for a class

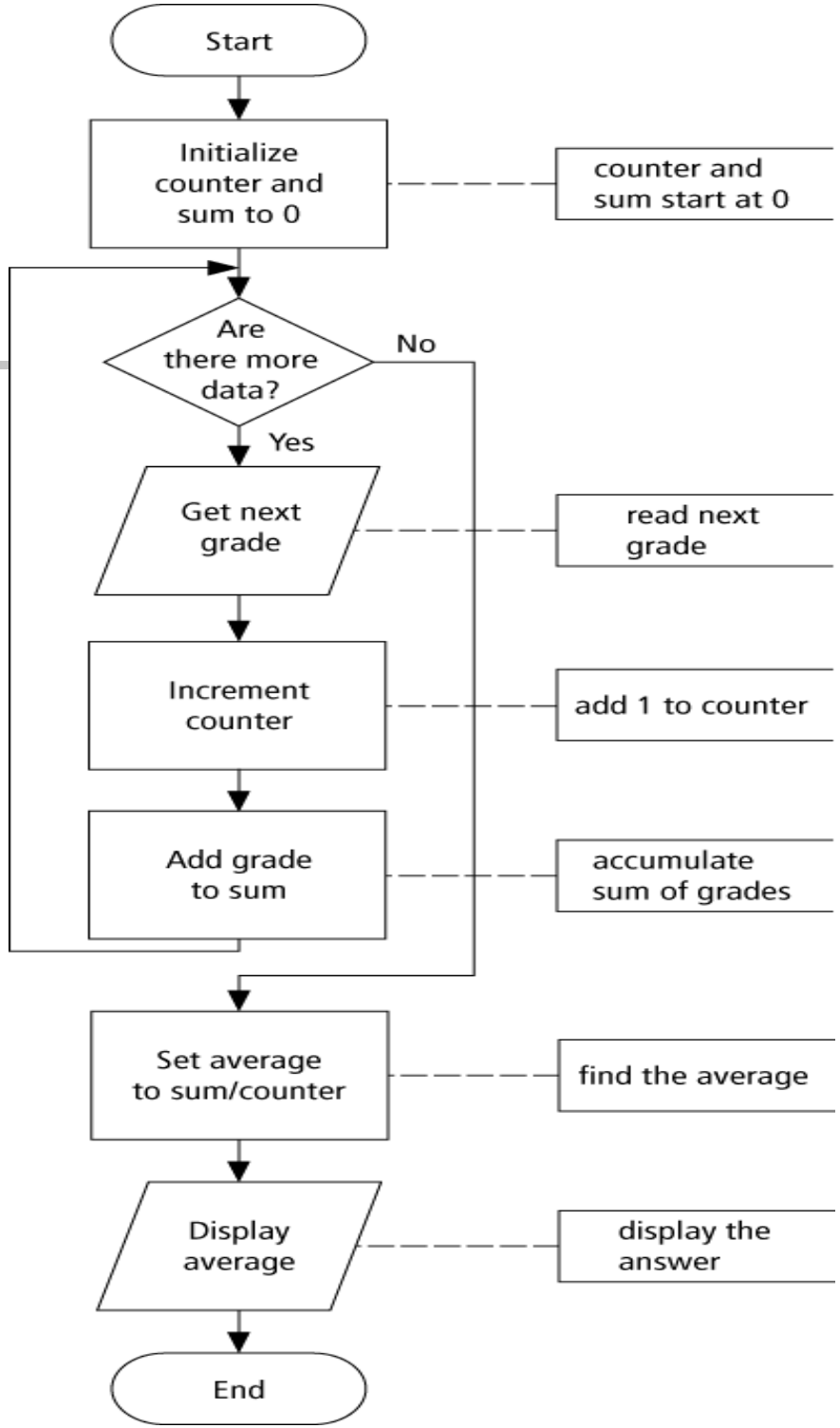
Discussion: The average grade equals the sum of all grades divided by the number of students

Input: Student grades

Processing: Find sum of the grades; count number of students; calculate average

Output: Average grade

Flowchart


Start
Initialize counter and sum to 0
counter and sum start at 0
Are there more data?
No
Yes
Get next grade
read next grade
Increment counter
add 1 to counter
Add grade to sum
accumulate sum of grades
Set average to sum/counter
find the average
Display average
display the answer
End

Pseudocode

Program: Determine average grade of a class

Initialize Counter and Sum to 0

Do While there are more data

    Get the next Grade

    Add the Grade to the Sum

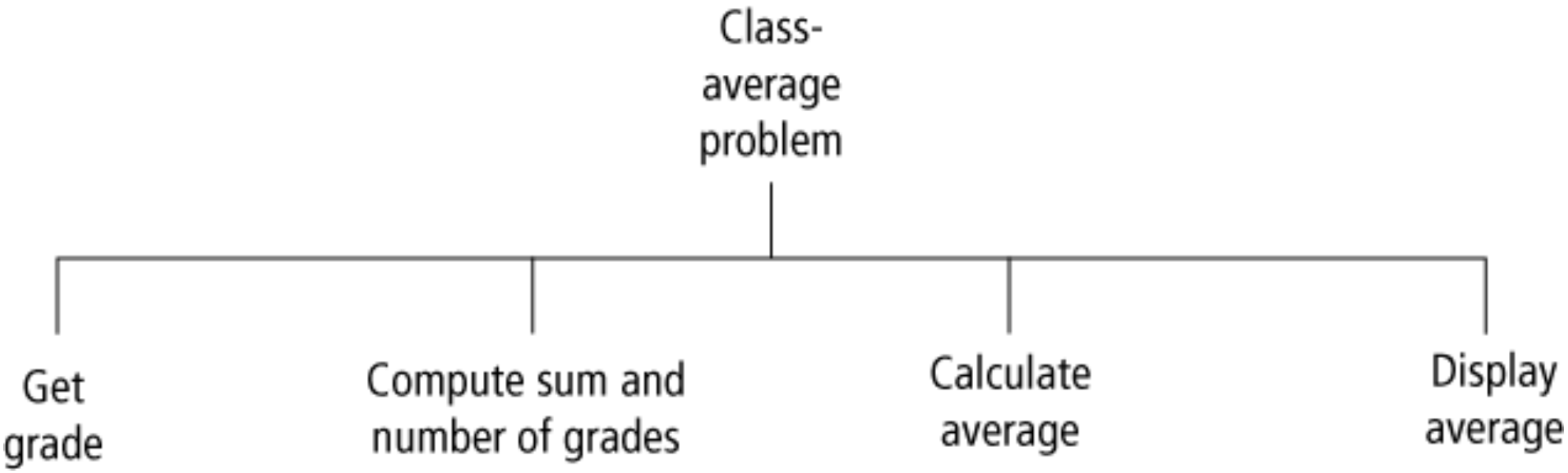
    Increment the Counter

Loop

Compute Average = Sum / Counter

Display Average

Hierarchy Chart



Comments

- When tracing a flowchart, begin at the start symbol and follow the flow lines to the end symbol.
- Testing an algorithm at the flowchart stage is known as **desk checking**.
- Flowcharts, pseudocode, and hierarchy charts are program planning tools that are in dependent of the language being used.

Tips and Tricks of Flowcharts

- Flowcharts are time-consuming to write and difficult to update
- For this reason, professional programmers are more likely to favor pseudocode and hierarchy charts
- Because flowcharts so clearly illustrate the logical flow of programs, they are a valuable tool in the education of programmers

1.4 Starting Python

Python is a general purpose programming language.
Python are now used in:

Google search engine, NASA, NYSE, …

Python 2 vs. Python 3

Python 3 is a newer version, but it is not backward compatible with Python 2. That means if you write a program using Python 2, it may not work on Python 3.

https://docs.python.org/

Python » 3.5.1 Documentation » modules | index

Download

Download these documents

Docs for other versions

Python 2.7 (stable)
Python 3.4 (stable)
Old versions

Other resources

PEP Index
Beginner's Guide
Book List
Audio/Visual Talks

Quick search

Go

Enter search terms or a module, class or function name.

Python 3.5.1 documentation

Welcome! This is the documentation for Python 3.5.1, last updated Mar 23, 2016.

Parts of the documentation:

What's new in Python 3.5?
or all "What's new" documents since 2.0

Tutorial
start here

Library Reference
keep this under your pillow

Language Reference
describes syntax and language elements

Python Setup and Usage
how to use Python on different platforms

Python HOWTOs
in-depth documents on specific topics

Installing Python Modules
installing from the Python Package Index & other sources

Distributing Python Modules
publishing modules for installation by others

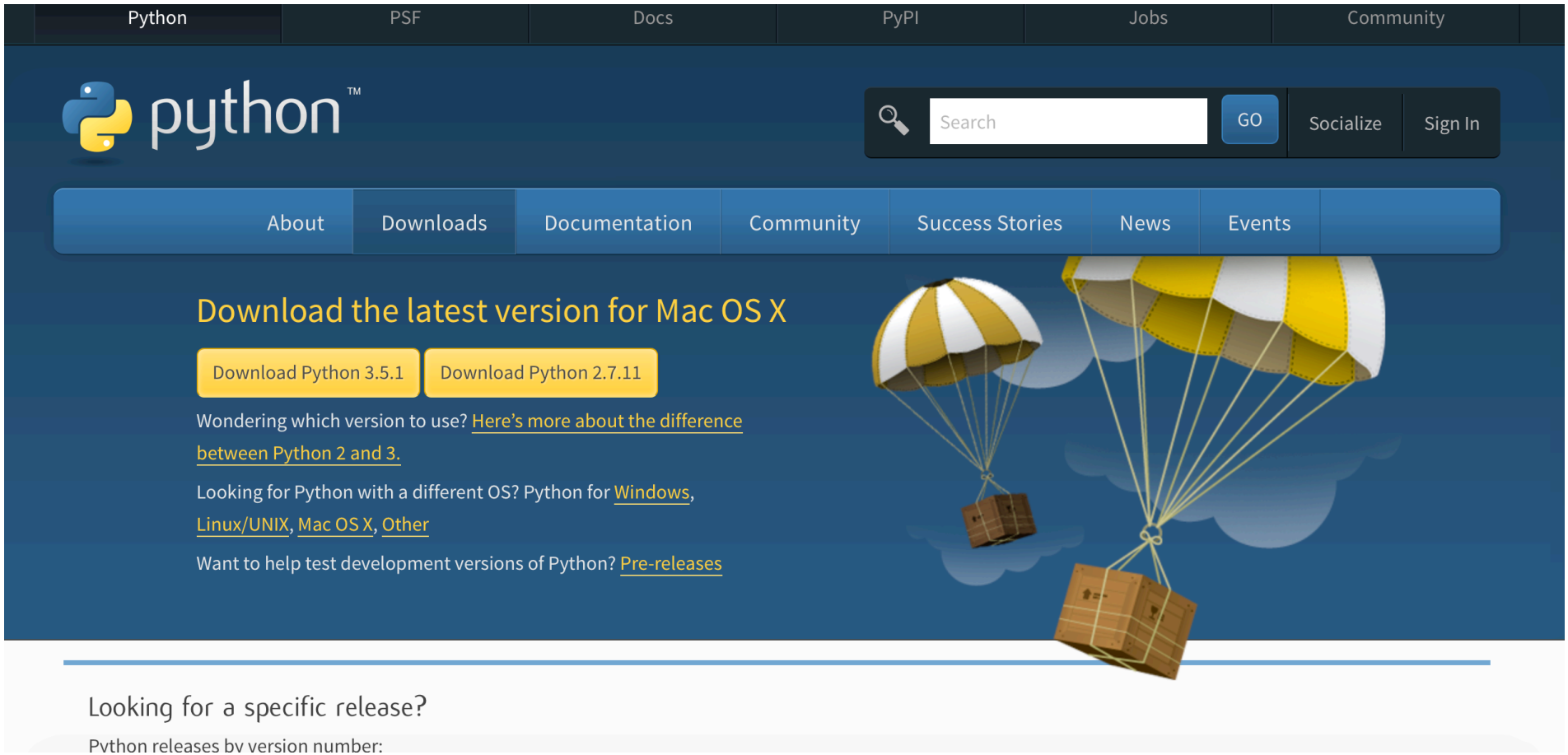
Extending and Embedding
tutorial for C/C++ programmers

Python/C API
reference for C/C++ programmers

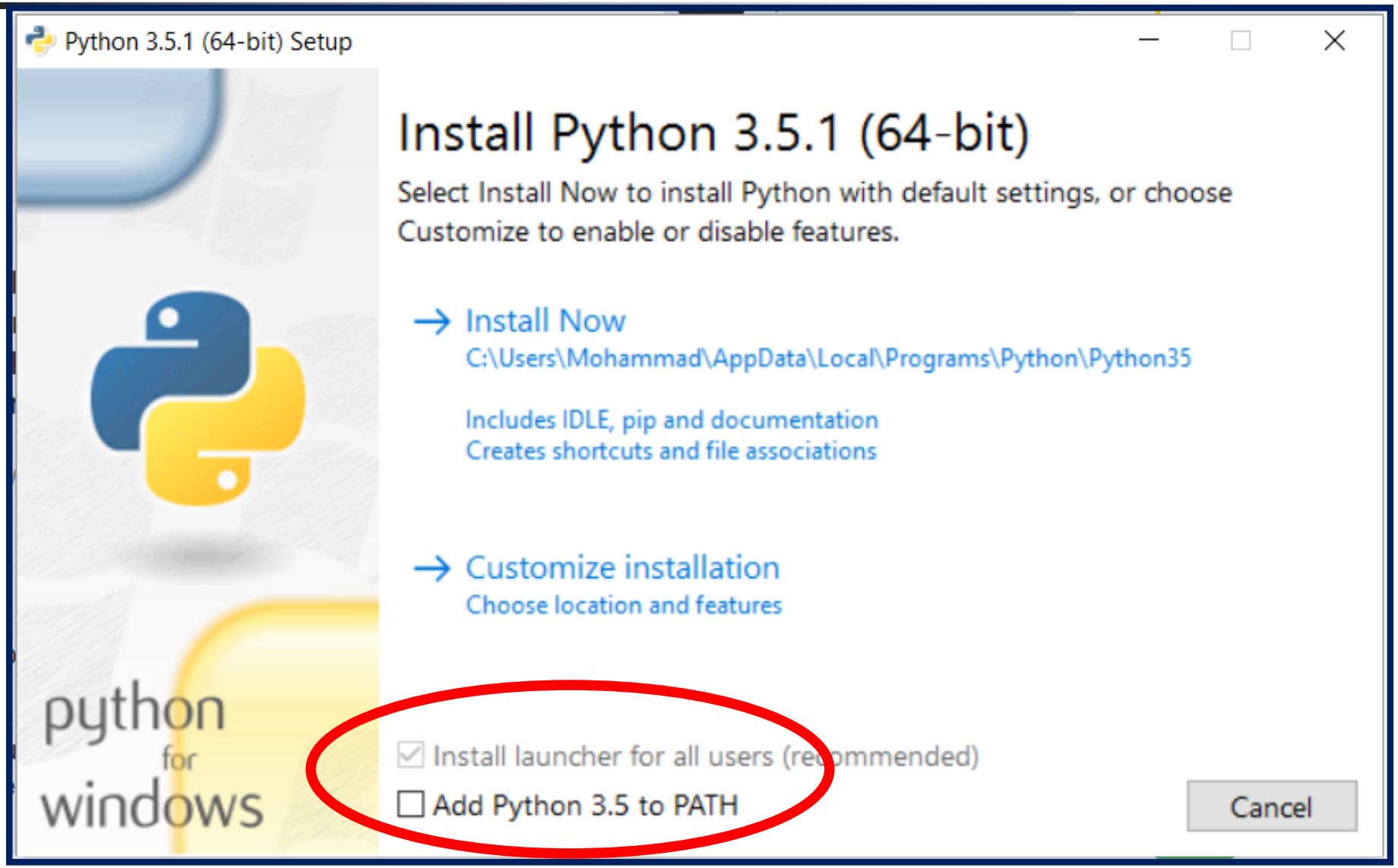
FAQs
frequently asked questions (with answers!)

Install Python (3.5.1)

- https://www.python.org/downloads/



For Windows Users



Install PyCharm (Community Version)

PyCharm

What's New Features Docs & Demos Buy **Download**

Download PyCharm

OS X WINDOWS LINUX

Professional

Full-featured IDE
for Python & Web
development

DOWNLOAD

246 MB

Community

Lightweight IDE
for Python & Scientific
development

DOWNLOAD

173 MB

Version: 2016.1.4

Build: 145.1504

Released: May 25, 2016

System requirements

Installation
Instructions

Previous versions

C U next week ☺