Code Academy

Chapter One: Introduction

PROGRAMMING AND PROBLEM SOLVING

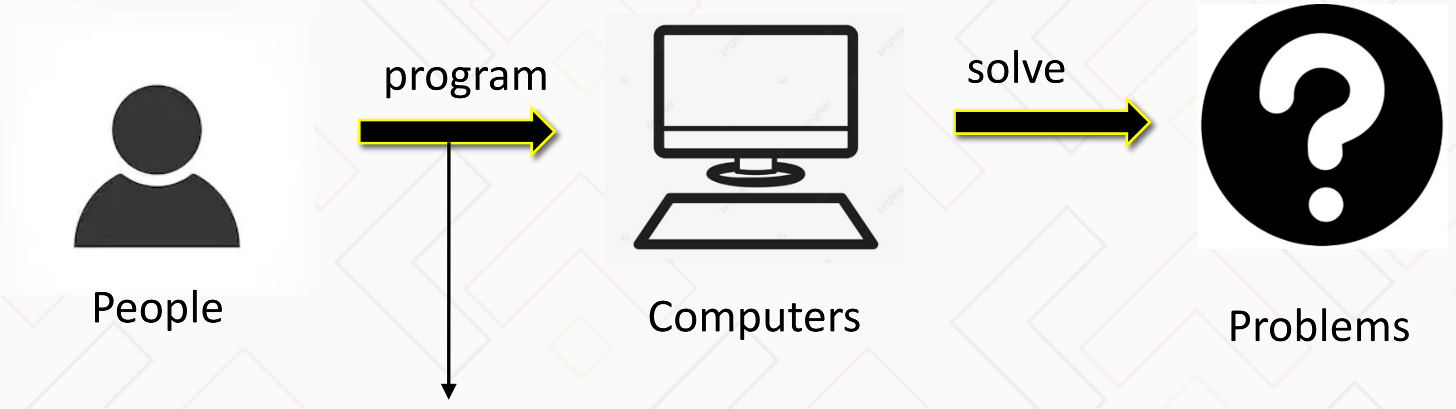
Code Academy

Computers are everywhere

- We want to put computers to solve problems in our lives, and if there isn't a ready made piece of software for us to use, we might have to program the computer ourselves in order to help us with our problem-solving.

Code Academy

Programming

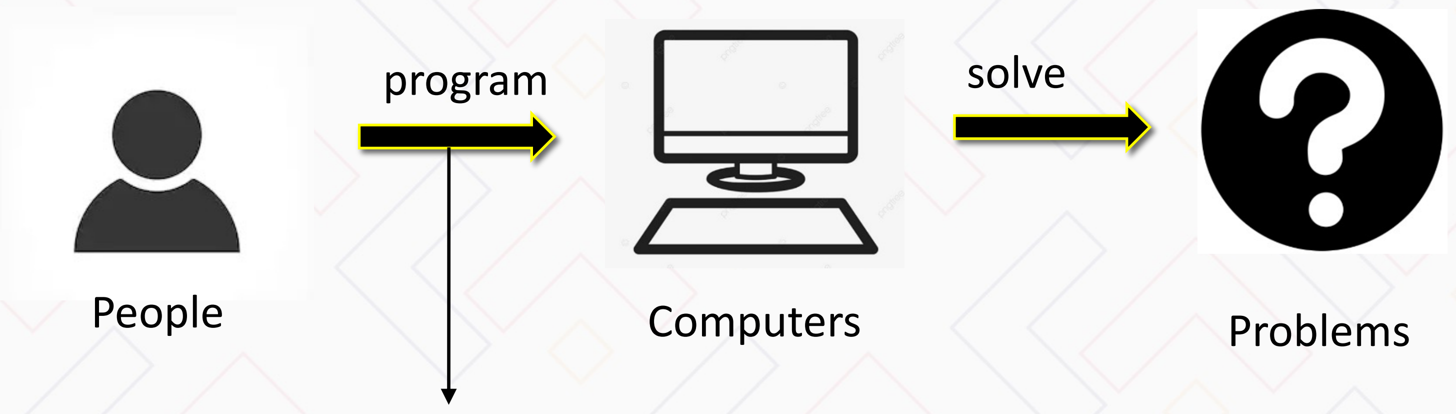


Computer program tells a computer the sequence of steps needed to solve a problem. The program is written in a very specific language (Python, C, Java,…)

However, Programming is not actually the first step!

Code Academy

Programming and Problem Solving

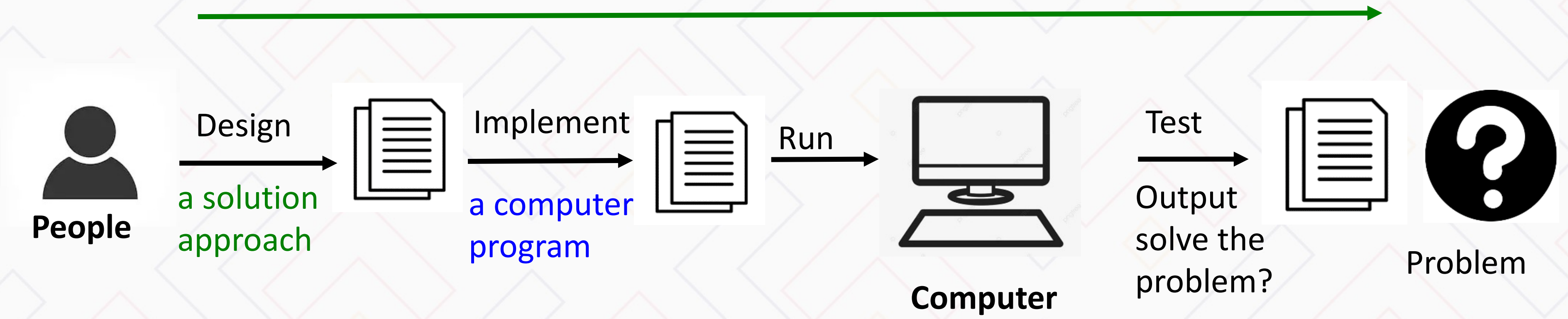


Computer program tells a computer the sequence of steps needed to solve a problem. The program is written in a very specific language (Python, C, Java,…)

- You have to engage in some problem-solving practices and thinking to set up the approach that we will later implement in the computer program.

Code Academy

Algorithmic Thinking



Design-> *Implement -> Run -> Test & Debug*

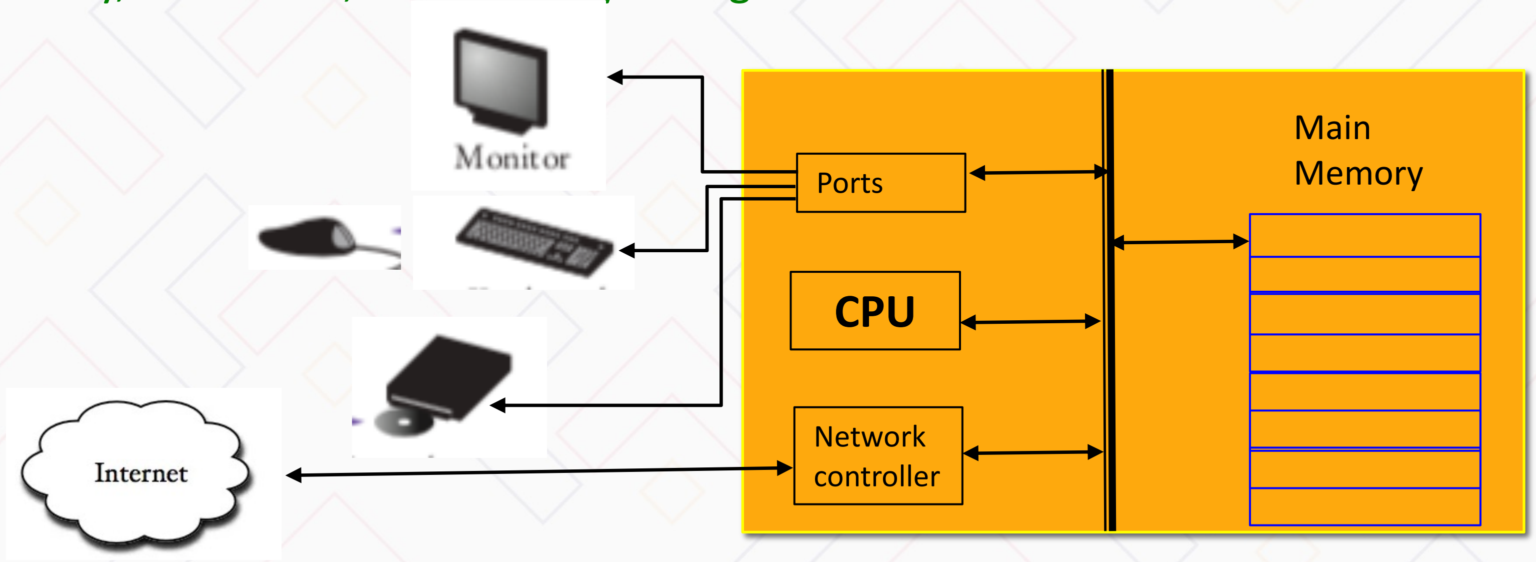
Code Academy

Computer Programs

- A computer **program** tells a computer the sequence of steps needed to complete a specific task
  - The program consists of a very large number of **primitive** (simple) **instructions**
  - Each program is designed to direct the computer to work on a specific task
- **Programming** is the act of **designing**, **implementing**, and **testing** computer programs
- If you want a computer to perform a task, you should start by writing an **algorithm** and then translate it to a computer program using a programming languages (e.g. Python).

Executing a Program

- Program instructions and data (such as text, numbers, audio, or video) are stored in **digital format**
- When a program is started, it is brought into **memory**, where the **CPU** can read it.
- The **CPU** runs the program *one instruction at a time*.
  - The program may react to user input.
  - The CPU reads data (including user input), modifies it, and writes it back to memory, the screen, or secondary storage.



Algorithms

A SHORT INTRODUCTION TO ALGORITHM DEVELOPMENT

Code Academy

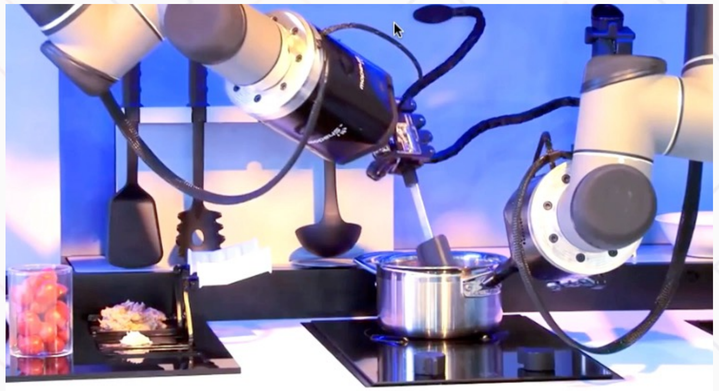
Introduction to Algorithms

- If you want a computer to perform a task, you start by writing an algorithm (mostly using a pen and paper)
- An **Algorithm** is:
  - a sequence (the order mattering) of actions to take to accomplish the given task
  - An algorithm is like a recipe; it is a set of instructions written in a sequence that achieves a goal
- Software developers write an algorithm before they attempt to write a computer program
- For this course we will ALWAYS write an algorithm for each problem
- Developing algorithms is a fundamental problem solving skill
  - It has uses in many fields outside of Computer Science

Code Academy

A Simple Example

- A simple algorithm to program a kitchen robot prepare a meal.
- E.g.: Prepare a drink of orange juice
  - Assumptions: for simplicity, the following are true:
    - You have a clean glass in the cabinet
    - You have orange juice in your refrigerator



>>> Run the kitchen robot video

- So one valid algorithm is:
  1. get a glass from your cabinet
  2. go to the refrigerator and get the orange juice container
  3. open the orange juice container
  4. pour the orange juice from the container into the glass
  5. put the orange juice container back in the refrigerator
  6. Serve the juice

Code Academy

Second Example: Selecting a Car

Problem Statement:

- You have the choice of buying two cars.
- One is more fuel efficient than the other, but also more expensive.
- You know the price and fuel efficiency (in miles per gallon, mpg) of both cars.
- You plan to keep the car for ten years.
- Which car is the better deal?

Code Academy

Developing the Algorithm

1. Determine the inputs and outputs
2. From the problem statement we know:
   - Car 1: Purchase price, Fuel Efficiency
   - Car 2: Purchase price, Fuel Efficiency
   - Price per gallon = $4.00
   - Annual miles driven= 15,000
   - Length of time = 10 years
3. For each car we need to calculate:
   - Annual fuel consumed for each car
   - Annual fuel cost for each car
   - Operating cost for each car
   - Total cost of each Car
4. Then we select the car with the lowest total cost

Code Academy

Translating the Algorithm to pseudocode

1. Break down the problem into smaller tasks

   - "Calculate total cost" for each car

   - To calculate the total cost for each year we need to calculate the operating cost
     - The operating cost depends on the annual fuel cost
     - The annual fuel cost is the price per gallon * the annual fuel consumed
     - The annual fuel consumed is the annual miles drive / fuel efficiency

   - total cost = purchase price + operating cost

2. Describe each subtask as pseudocode

The Pseudocode

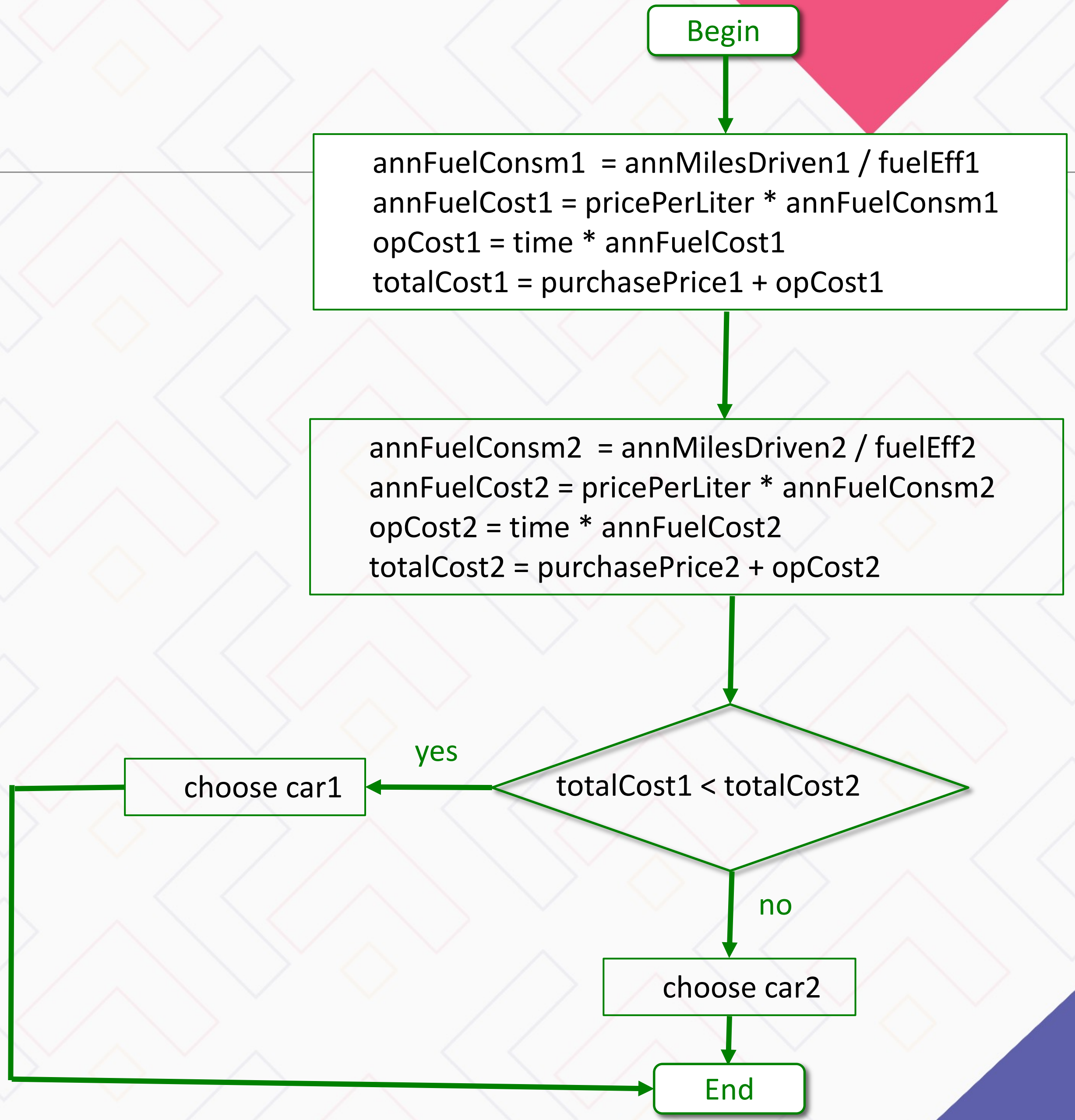
```
1.    For each car compute the **total cost**:
        annual fuel consumed  = annual miles driven / fuel efficiency
        annual fuel cost = price per gallon * annual fuel consumed
        operating cost = Length of time * annual fuel cost
        total cost = purchase price + operating cost

2.    if total cost1 < total cost2
          choose car1
      else
          choose car2
```

- In computer science, **pseudocode** is a plain language description of the steps in an algorithm. Pseudocode resembles a normal programming language, but is intended for human reading rather than machine reading.

Code Academy

The Flowchart

- A **flowchart** is a type of diagram that represents a workflow or process.
- A flowchart is a diagrammatic representation of an algorithm
- The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.


Begin
annFuelConsm1 = annMilesDriven1 / fuelEff1
annFuelCost1 = pricePerLiter * annFuelConsm1
opCost1 = time * annFuelCost1
totalCost1 = purchasePrice1 + opCost1
annFuelConsm2 = annMilesDriven2 / fuelEff2
annFuelCost2 = pricePerLiter * annFuelConsm2
opCost2 = time * annFuelCost2
totalCost2 = purchasePrice2 + opCost2
totalCost1 < totalCost2
yes
choose car1
no
choose car2
End

Code Academy

Algorithmic Thinking

- The thinking that we do before and as we start writing a computer program

1. Problem Identification
2. Decomposition
3. Pattern Recognition
4. Abstraction

Code Academy

Algorithmic Thinking

- The thinking that we do before and as we start writing a computer program

1. **Problem Identification**

   - How do we know when a problem is a good candidate for using a computer to generate a solution?

   - What's the big problem you're trying to solve, and can you state this in an adequate **problem statement**?

   - What kind of information or **data** do we have available that could help us or contribute to a problem-solving approach?

   - What are the **assumptions**?

Code Academy

Algorithmic Thinking

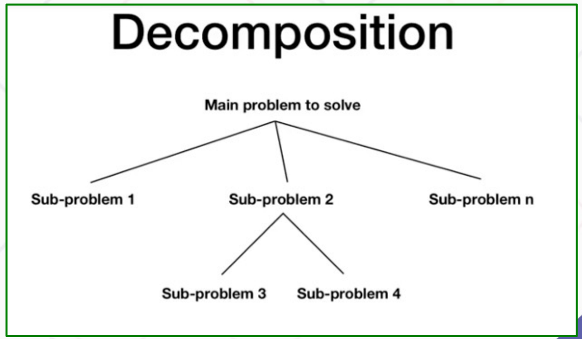
- The thinking that we do before and as we start writing a computer program

2. Problem Decomposition

- Break our large problem down into smaller and smaller **sub-problems**
- Solve those specific sub-problems.
- Have these different solutions all contribute to solving the large problem.

Solving the smaller pieces can help us arrive at a solution for our original problem statement.

Hint: Use flowchart, and tree structure



Code Academy

Algorithmic Thinking

- The thinking that we do before and as we start writing a computer program

3. Pattern recognition

- Have you seen solutions to similar problems that you could use here? Or can you think of similar problems that have been solved before that can help you address these problems?

- How is this problem the same or different from other problems you've identified or addressed in the past? And can you use those similarities to help arrive at a solution?

Finding similarities, finding some problems that are similar patterns that we have solved in the past can help us solve the smaller decomposed problems more efficiently.

Code Academy

Algorithmic Thinking

- The thinking that we do before and as we start writing a computer program

4. Abstraction

- Are all elements of the problem really as important as we may have thought at the beginning?

- Identify non-essential aspects of our problem that we might be able to **ignore**

- So that we can concentrate on the necessary aspects of the problem we're trying to solve.

Code Academy

Case Study: How to detect suspicious behavior on an airport surveillance videos

>>> Run the video surveillance

Reflect

>>> What types of suspicious behavior would you look for at an airport?

>>> How do you think computers can help us identify suspicious behavior?

- Engage in problem solving thinking, then

- Share your thoughts on the Virtual Classroom Forum…

Code Academy

Engage in Computational Thinking: How to detect suspicious behavior on an airport surveillance videos

1. **Problem Identification:**
   - Is the problem a good candidate for using a computer to generate a solution?

2. **Decomposition:**
   - How would you break down the problem into smaller sub-problems?

3. **Pattern Recognition:**
   - Are there related solutions to build on?

4. **Abstraction:**
   - Are there non-essential aspects of our problem that might be ignored

Code Academy

Programming and Problem Solving

- The main goal of this course is to help you develop
  - your computational thinking and
  - programming skills to solve problems

- Problems solving has uses in many fields outside of Computer Science