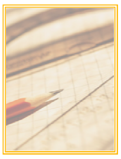
CHAPTER 1:
INTRODUCTION TO COMPUTER SCIENCE

Introduction to Computer Science Using Ruby

What is Computer Science?

- The study of **algorithm** design and implementation
  - **Algorithm**: the description of the steps required to solve a problem
  - The steps are expressed by **instructions** to the computer

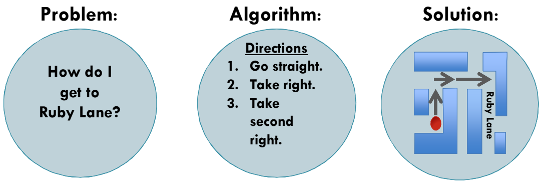
What is Computer Science?



- **Computer program**: the sequence of all the instructions needed for the algorithms
- **Programming language**: the tool to specify the computer program
  - Examples: Ruby, C, C++, Python, Java

What is Computer Science?

- The study of algorithms helps to **create solutions** that yield correct results in a **timely manner**



Problem:

How do I get to Ruby Lane?

Algorithm:

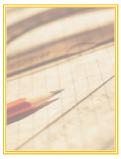
Directions
1. Go straight.
2. Take right.
3. Take second right.

Solution:

Application Development

- Computers follow instructions, so **all details must be specified.**
  - Example: “Spread peanut butter on one slice of bread, and jelly on another slice of bread. Then put the slices of bread together.”
    - This is enough information for a human to make a peanut butter-jelly sandwich, but not enough information for a computer.

A Basic Approach to Problem Solving



- Step 1: Understand the problem
- Step 2: Write out the problem in plain language
- Step 3: Translate the plain language into programming language
- Step 4: Test the code in the computer

Step 1: Understand the Problem

- Try to answer all the questions about the problem
- Look for the goals, data available, and results to be derived

- **Example**: A program that stores a list of names
  - How many names will be stored?
  - Do first and last names need to be stored separately?
  - Are middle names needed?
  - What is the maximum length that a name can be?

Step 2: Write out the Problem in English

- Determine the steps necessary to go from **data** to **results**
- Make an outline of the solution

- **Example**: A program that stores a list of names
  1. Ask for the first name.
  2. Store the first name.
  3. Ask for the last name.
  4. Store the last name.
  5. Ask for the middle initial.
  6. Store the middle initial.

Step 3: Translate the English into Programming Language

- Write the specification based on the English version
- Include **comments**
  - Non-executable documentation aimed to help the user

- **Example:** A program that stores a list of names

```
puts "Enter first name: ” # Ask for the first name
first_name = gets # Store the first name
puts "Enter last name: ” # Ask for the last name
last_name = gets # Store the last name
puts "Enter middle initial: ” # Ask for middle initial
middle_initial = gets # Store the middle initial
```

(c) 2012 Ophir Frieder et al

Step 4: Test the Code in the Computer

- See if the code runs properly
- Test the code in sections as your write

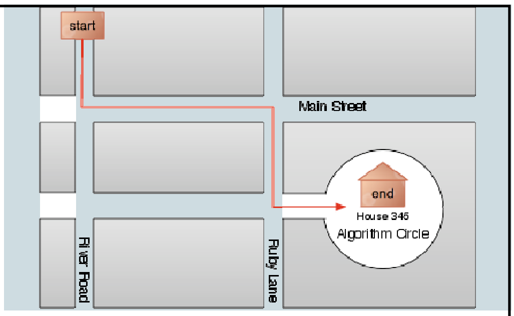
(c) 2012 Ophir Frieder et al

Algorithms

- Step-by-step methods for solving problems
- Describe the process, but not each and every minutia

- **Example Algorithm for Directions:**
  1. Start going South on River Road.
  2. Turn left (East) on Main Street.
  3. Take a right (South) on Ruby Lane.
  4. Turn left (East) toward Algorithm Circle.
  5. Continue until you come to 345 Algorithm Circle.

(c) 2012 Ophir Frieder et al

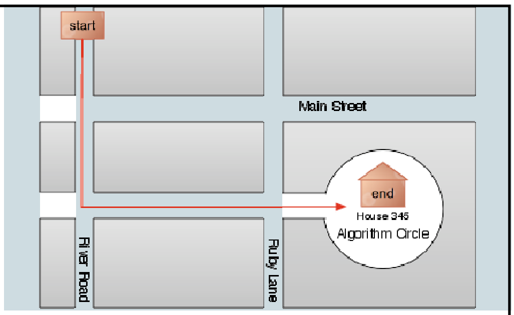


Example Algorithm for Directions (Figure 1.2)

(c) 2012 Ophir Frieder et al

Algorithms

- Efficient algorithms reduce the number of steps needed
- **Improved Algorithm for Directions:**
  1. Start going South on River Road.
  2. Turn lest (East) one block South of Main Street on to Algorithm Circle.
  3. Continue until you come to 345 Algorithm Circle.



Improved Algorithm for Directions (Figure 1.3)

Summary

Algorithms: step-by-step methods for solving problems

- Computer science uses algorithms to solve problems
- The computer needs every detail to solve the problem
- Efficiency should be considered

Basic Steps for Computer Science Programs:

- Step 1: Understand the problem.
- Step 2: Write out a plain language (or another pseudo-code) solution.
- Step 3: Translate the plain language or pseudo-code into code.
- Step 4: Test the code in the computer.