# CMSC201
# Computer Science I for Majors

## Lecture 05 – Algorithmic Thinking

# Last Class We Covered

- Decision structures

- One-way (using **`if`**)
- Two-way (using **`if`** and **`else`**)
- Multi-way (using **`if`**, **`elif`**, and **`else`**)

- Nested decision structures

# Any Questions from Last Time?

# Today's Objectives

- To practice thinking algorithmically
- To understand and be able to implement proper program development
  - To learn more about "bugs"

- To get practice with decision structures
- (Lots of practice)

# What is an Algorithm?

- Steps used to solve a problem

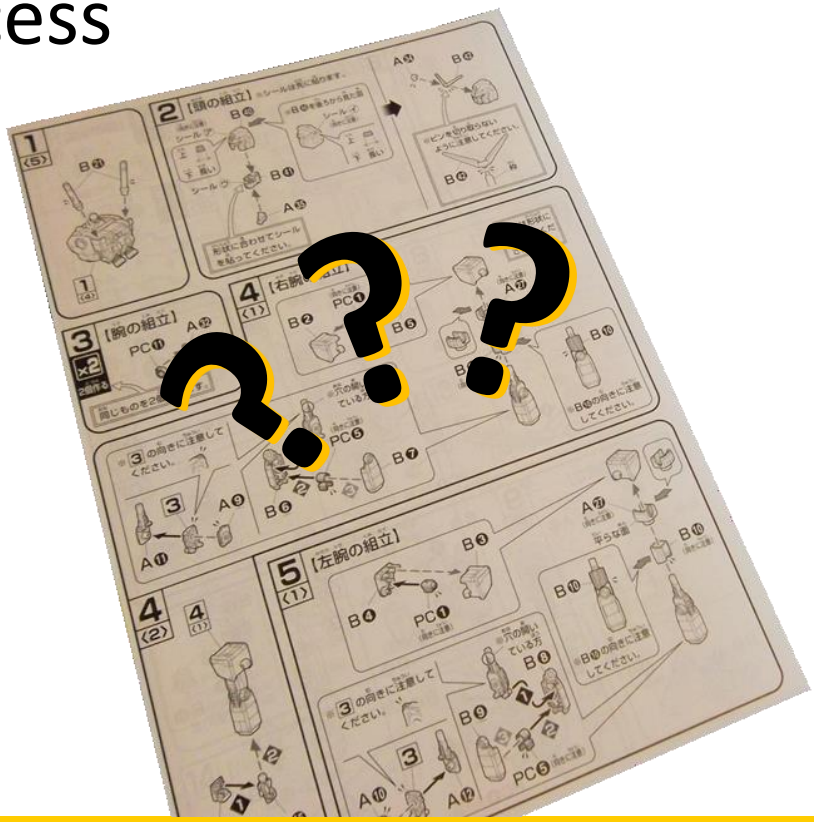
- Problem must be
  - Well defined
  - Fully understood by the programmer

- Steps must be
  - Ordered
  - Clear
  - Complete

# Algorithmic Thinking

- Algorithms are an ordered set of clear steps that fully describes a process

- Examples from real life?
  - Recipes
  - Driving directions
  - Instruction manual (IKEA)
    - (maybe not so much)



# Developing an Algorithm

# Program Development

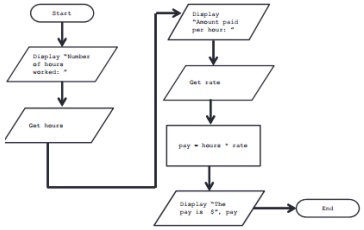
1. Understand the problem
2. Represent your solution (your algorithm)
   - Pseudocode
   - Flowchart
3. Implement the algorithm in a program
4. Test and debug your program

# Step 1: Understanding the Problem

- Input
  - What information or data are you given?
- Process
  - What must you do with the information/data?
  - **This is your algorithm!**
- Output
  - What are your deliverables?

# Step 2: Represent the Algorithm

- Can be done with flowchart or ***pseudocode***
- Flowchart




  - Symbols convey different types of actions
- Pseudocode
  - A cross between code and plain English
- One may be easier for you – use that one

# Steps 3 and 4: Implementation and Testing/Debugging

- Implementing and testing/debugging your program are two steps that go hand in hand

- After implementing, you must test it
- After discovering errors, you must find them
  - Once found, you must fix them
  - Once found and fixed, you must test again

# Development Example: Weekly Pay

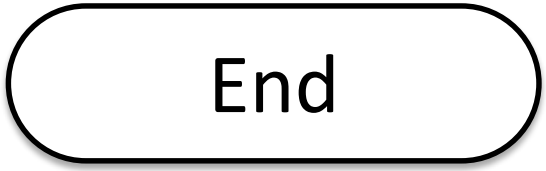
- Create a program to calculate the weekly pay of an hourly employee
  - What is the input, process, and output?

- Input: pay rate and number of hours
- Process: multiply pay rate by number of hours
- Output: weekly pay

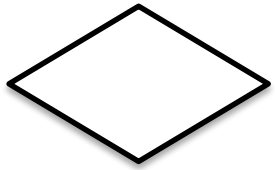
# Flowchart Symbols


Start Symbol

Input/Output




End Symbol

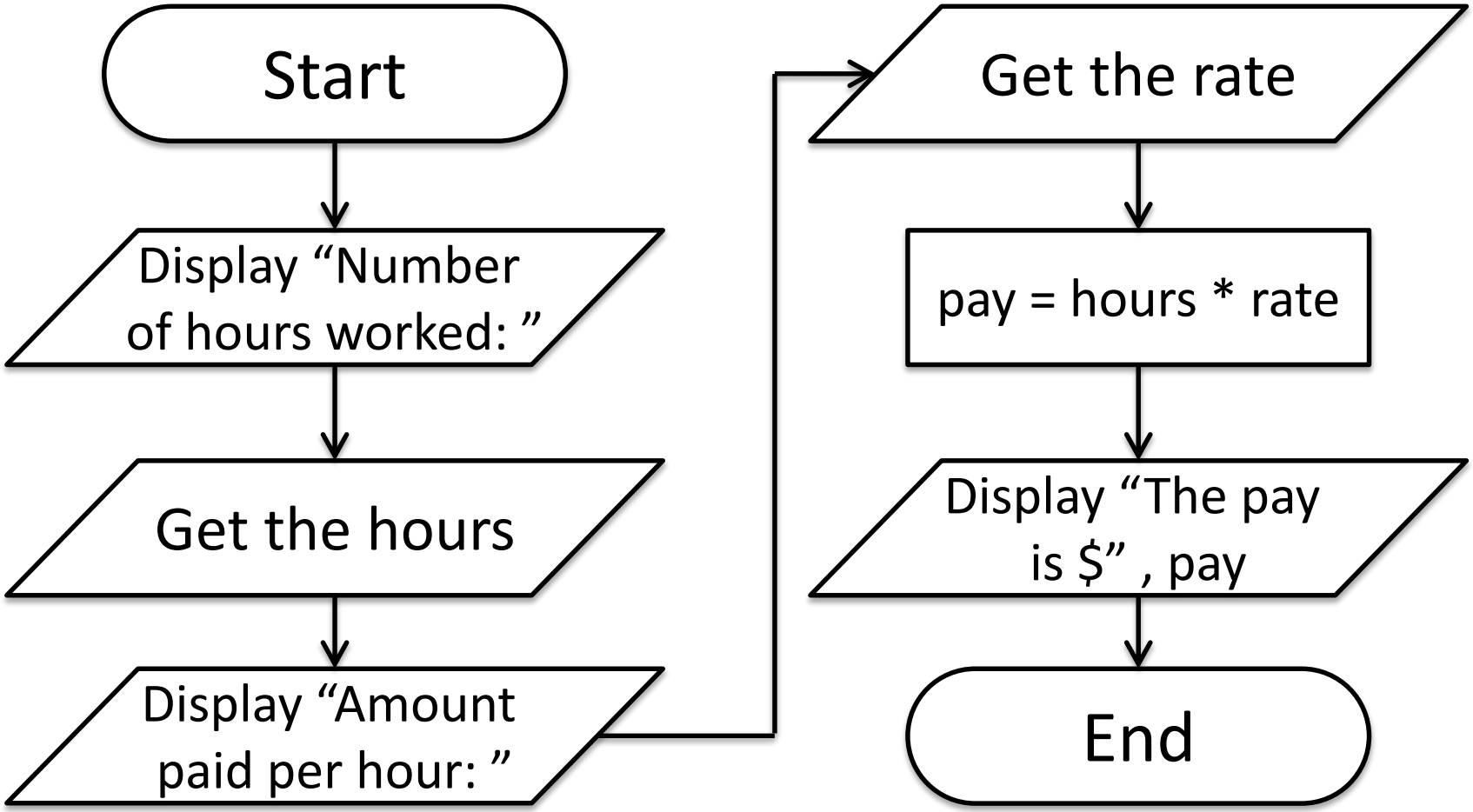


Decision Symbol

Data Processing Symbol

Flow Control Arrows

# Step 2A: Flowchart



# Step 2B: Pseudocode

- Start with a plain English description, then…

```
1. Display "Number of hours worked: "
2. Get the hours
3. Display "Amount paid per hour: "
4. Get the rate
5. Compute pay = hours * rate
6. Display "The pay is $" , pay
```

As a general rule, each line of pseudocode should start with a verb stating the action being taken. ("Get," "decide," "display," etc.)

# Algorithms and Language

- Notice that developing the algorithm didn't involve any Python at all
  - Only pseudocode or a flowchart was needed
  - An algorithm can be coded up in any language
- All languages share certain tools that can be used in your algorithms
  - For example, ***control structures*** and ***expressions***

# Exercise: Are Dogs Good?

- Ask the user if a dog is a good dog
- Print out one response for “yes”
- Print out a different response for any other answer

# Debugging

# A Bit of History on "Bugs"

Rear Admiral Grace Hopper

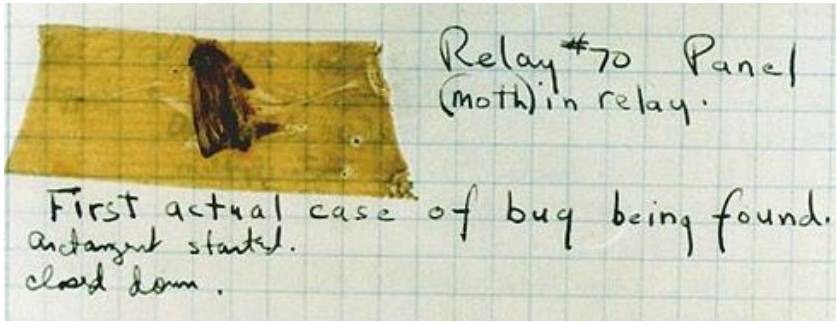
- US Navy lab (Sep 1947)
- Grace Hopper and her colleagues were working on the Harvard Mark II
  - Instructions read one at a time from a tape
- Or trying to… it wasn't working right

# A Bit of History on "Bugs"

Mark II, general view of calculator frontpiece, 1948.

- Mark II was a LARGE machine that took up an entire room
  - You could open each panel and look inside

- They found a moth inside the machine
  - Taped the bug into their log book



# Errors ("Bugs")

- Two main classifications of errors

- Syntax errors
  - Prevent Python from understanding what to do
- Logical errors
  - Cause the program to run incorrectly, or to not do what you want

# PB&J Using Exact Instructions

- "You're not even making any sense! He's already ruined it on purpose, he knows how to make one."

- Watch the video here
  - (Image from Josh Darnit's Exact Instructions Challenge)

# Syntax Errors

- "Syntax" is the set of rules followed by a computer programming language
  - Similar to grammar and spelling in English
- Examples of Python's syntax rules:
  - Keywords must be spelled correctly

    **`True`** and **`False`**, not **`Ture`** or **`Flase`** or **`Truu`**
  - Quotes and parentheses must be closed in order:

```
("open and close")
```

# Syntax Error Examples

- Find the syntax errors in each line of code below:

```
1    prnit("Hello")
2    print("What"s up?")
3    print("Aloha!)
4    print("Good Monring")
```

# Syntax Error Examples

- Find the syntax errors in each line of code below:

```
1   prnit("Hello")
2   print("What"s up?")
3   print("Aloha!)
4   print("Good Monring")
```

not actually a syntax error

# Syntax Error Examples

- Find the syntax errors in each line of code below:

```
1    prnit("Hello")
2    print("What"s up?")
3    print("Aloha!)
4    print("Good Monring")
```

The syntax highlighting in emacs can often help you see where the errors are

# Logical Errors

- Logical errors don't bother Python at all… they only bother you!

- Examples of logical errors:
  - Using the wrong value for something

```
currentYear = 2013
```

  - Doing steps in the wrong order
    - "Place pan in the oven. Preheat oven to 350. Pour batter into pan, spreading evenly."

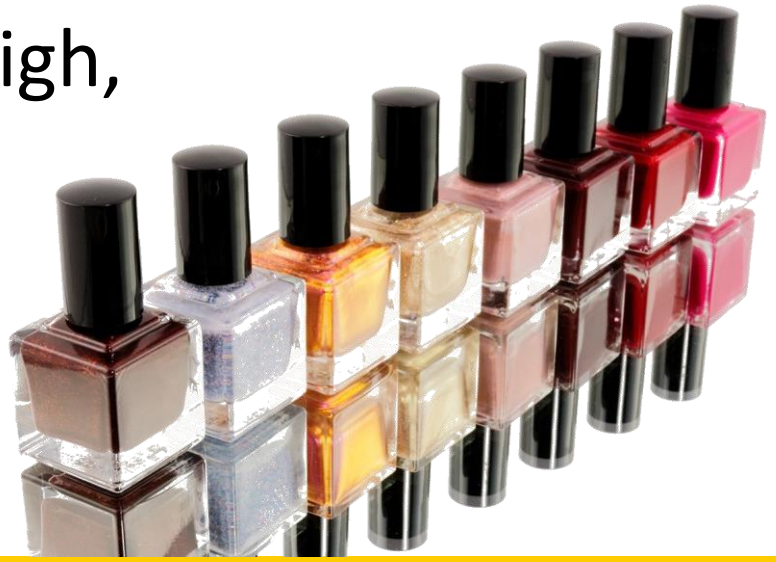
# Comments in Debugging

- Comments are often used to convey what your program is doing
  - If there is a bug, however, your code may not actually be accomplishing that task
- Comments are very useful when debugging, because they separate intent from actuality
  - "Is your code working?" and "Is your code doing what it's supposed to do?" are very different questions

# Practicing Decision Structures

# Exercise: Nail Polish

- Dr. Gibson has a LOT of nail polish
- Write a game where the user guesses how many bottles she has, and tell them whether their guess was high, low, or correct
- What info do you need?
  - (She has 322 bottles)



# Exercise: Moving on to CMSC 202

- Ask the user their major and the grade they earned in CMSC 201
  - Print out whether they can move on to CMSC 202 next semester
- If they're a CMSC or CMPE major
  - They need an A or a B
- Otherwise
  - They need an A, B, or a C

# Daily emacs Shortcut

- **CTRL+S**
  - Allows you to search within a file
  - (To remember: S stands for “search”)
  - Hit CTRL+S, then type in what you want to find
  - Hit CTRL+S again to find the next occurrence
  - If you reach the end of the file and want to start back at the beginning, hit CTRL+S again
  - Use any movement (arrows, etc.) to exit

# Announcements

- HW 1 is out on Blackboard now
  - Must complete the Syllabus and Course Website Quiz to see it
  - Due by Friday (February 15th) at 11:59:59 PM

- Pre Lab 4 Quiz will come out Friday @ 10 AM
  - Must be completed by 10 AM Monday morning

# Image Sources

- IKEA instructions (adapted from):
  - https://www.flickr.com/photos/girlinblack/6697086037
- Three dogs:
  - https://pixabay.com/p-984015/
- Rear Admiral Grace Hopper:
  - https://commons.wikimedia.org/wiki/File:Grace_Hopper.jpg
- Mark II:
  - http://amhistory.si.edu/archives/images/d8324-1.jpg
- Notebook bug (adapted from):
  - https://commons.wikimedia.org/wiki/File:H96566k.jpg
- Computer bug:
  - https://pixabay.com/p-1296767/
- Nail polish (adapted from):
  - https://pixabay.com/p-870857/
- Question mark man:
  - https://pixabay.com/p-1019993/