

# CMSC201

## Computer Science I for Majors

### Lecture 02 – Algorithmic Thinking

# Last Class We Covered

- Syllabus
  - Grading scheme, expectations, etc.
  - Academic Integrity Policy
    - (Collaboration Policy)
- Binary numbers
  - Converting between binary and decimal
- Algorithmic thinking
  - Making sandwiches for aliens

Any Questions from Last Time?

# Today's Objectives

- To practice thinking algorithmically
- To understand and be able to implement proper program development
- To start learning about control structures
- To be able to express an algorithm using a flow chart

# What is an Algorithm?

- Steps used to solve a problem
- Problem must be
  - Well defined
  - Fully understood by the programmer
- Steps must be
  - Ordered
  - Unambiguous
  - Complete

# 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?

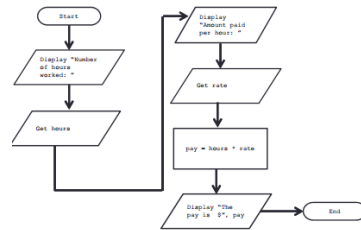


# “Weekly Pay” Example

- 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

# Step 2: Represent the Algorithm

- Can be done with flowchart or *pseudocode*



- Flowchart

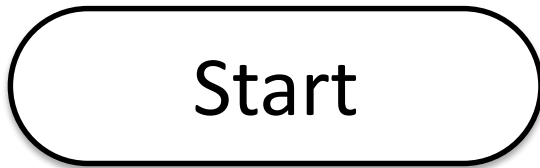
- Symbols convey different types of actions

- Pseudocode

- A cross between code and plain English

- One may be easier for you – use that one

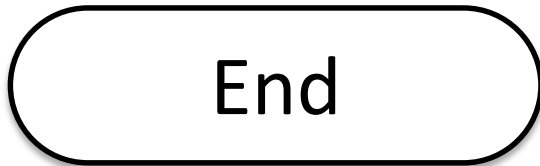
## Flowchart Symbols



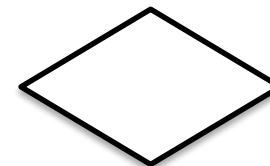
Start Symbol



Input/Output



End Symbol



Decision Symbol

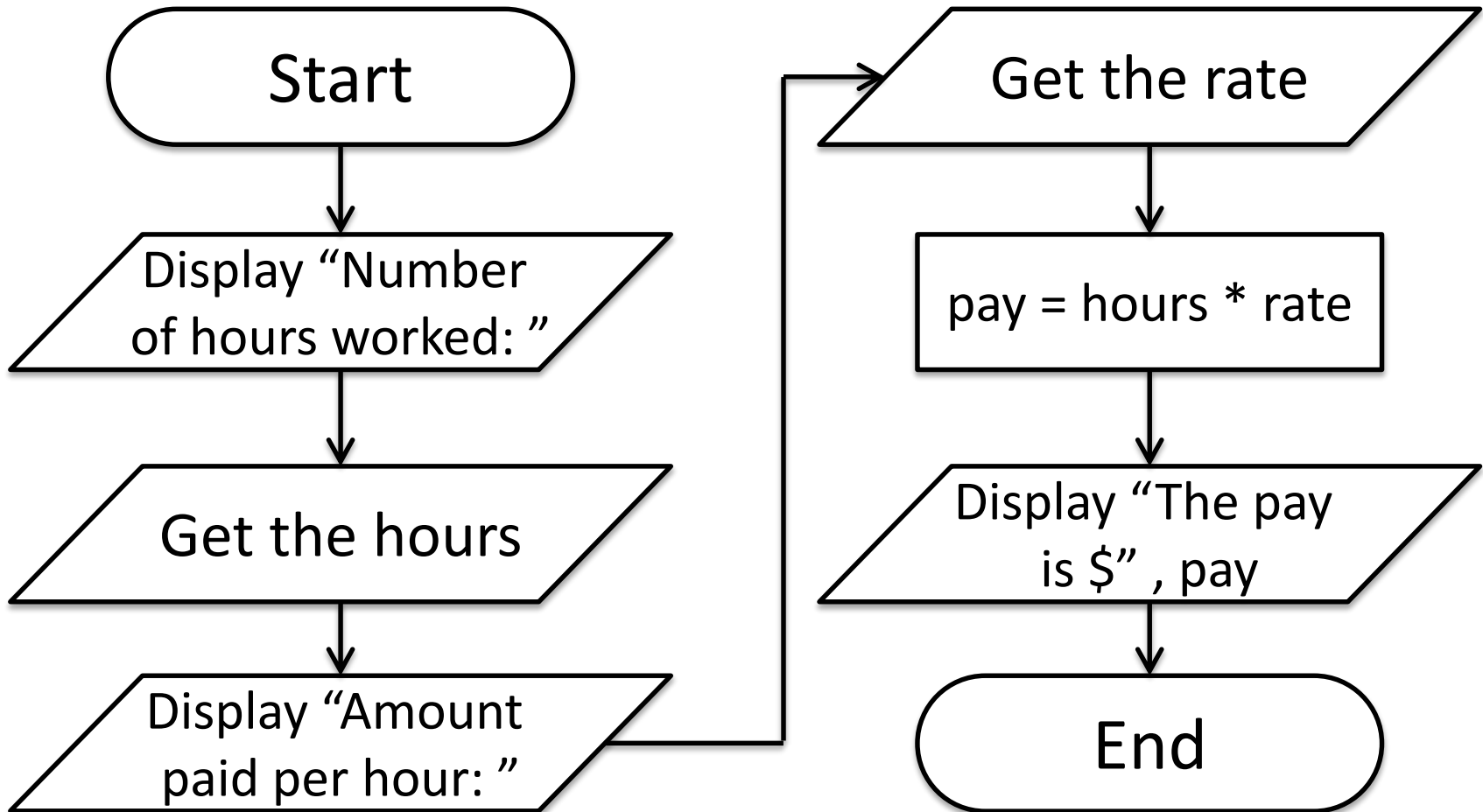


Data Processing Symbol



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  $\text{pay} = \text{hours} * \text{rate}$
  6. Display "The pay is \$" , pay

# Steps 3 and 4: Implementation and Testing/Debugging

- We'll cover implementation in detail next class
- Testing and debugging your program involves identifying errors and fixing them
  - We'll talk about this later today

# 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 in any language
- All languages have 3 important tools called ***control structures*** that we can use in our algorithms

# Control Structures



# Control Structures

- Structures that control how the program “flows” or operates, and in what order
- Sequence
- Decision Making
- Looping

# Sequence

- One step after another, with no branches
- Already wrote one for “Weekly Pay” problem
- What are some real life examples?
  - Dialing a phone number
  - Purchasing and paying for groceries

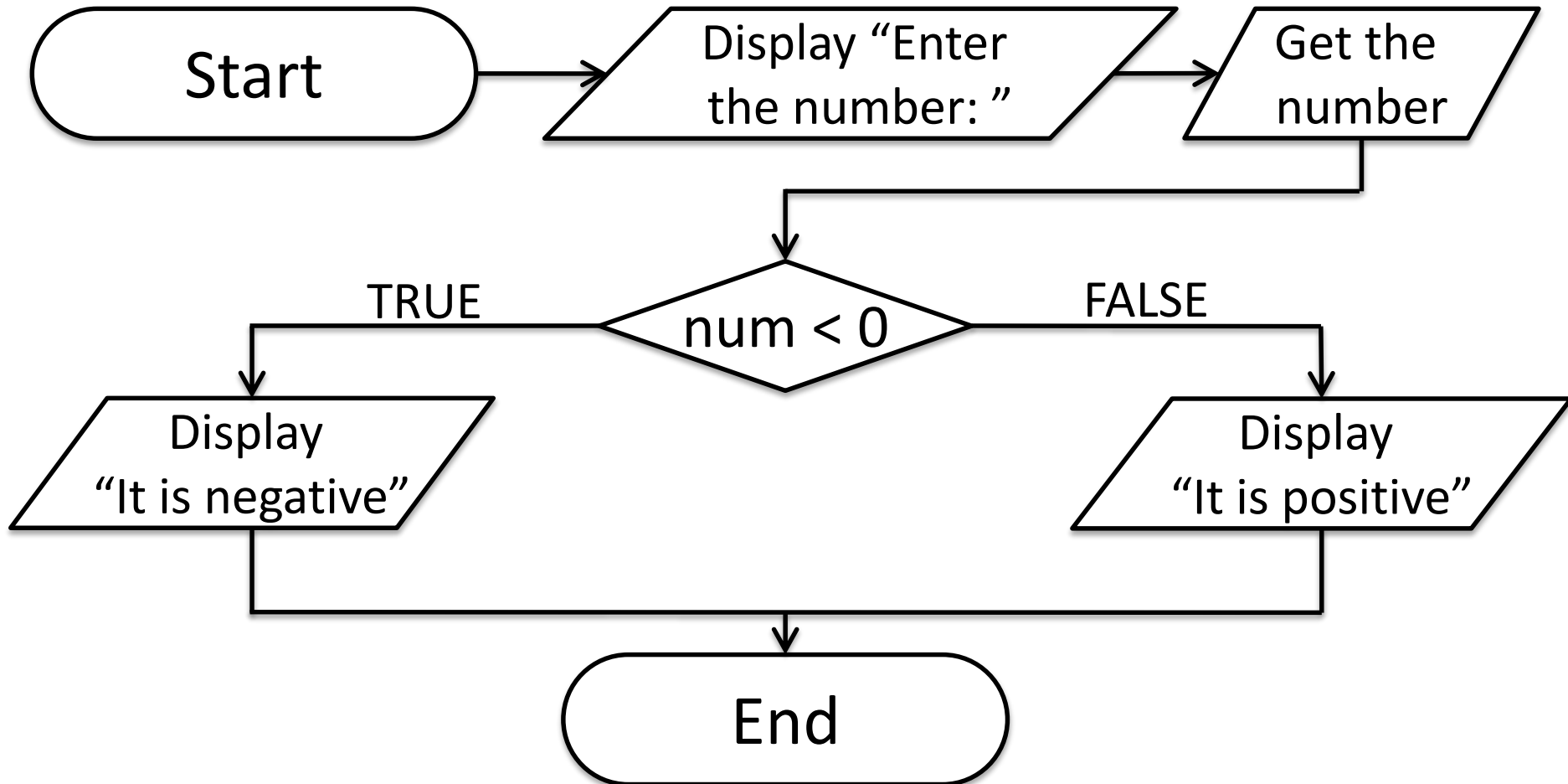
# Decision Making

- Selecting one choice from many based on a specific reason or condition
  - If something is true, do **A** ... if it's not, do **B**
- What are some real life examples?
  - Walking around campus (construction!)
  - Choosing where to eat lunch

# Decision Making: Pseudocode

- Answer the question “Is a number positive?”
    - Start with a plain English description
1. Display "Enter the number: "
  2. Get the number (call it num)
  3. If  $\text{num} < 0$
  4.     Display "It is negative"
  5. Else
  6.     Display "It is positive"

# Decision Making: Flowchart



# Looping

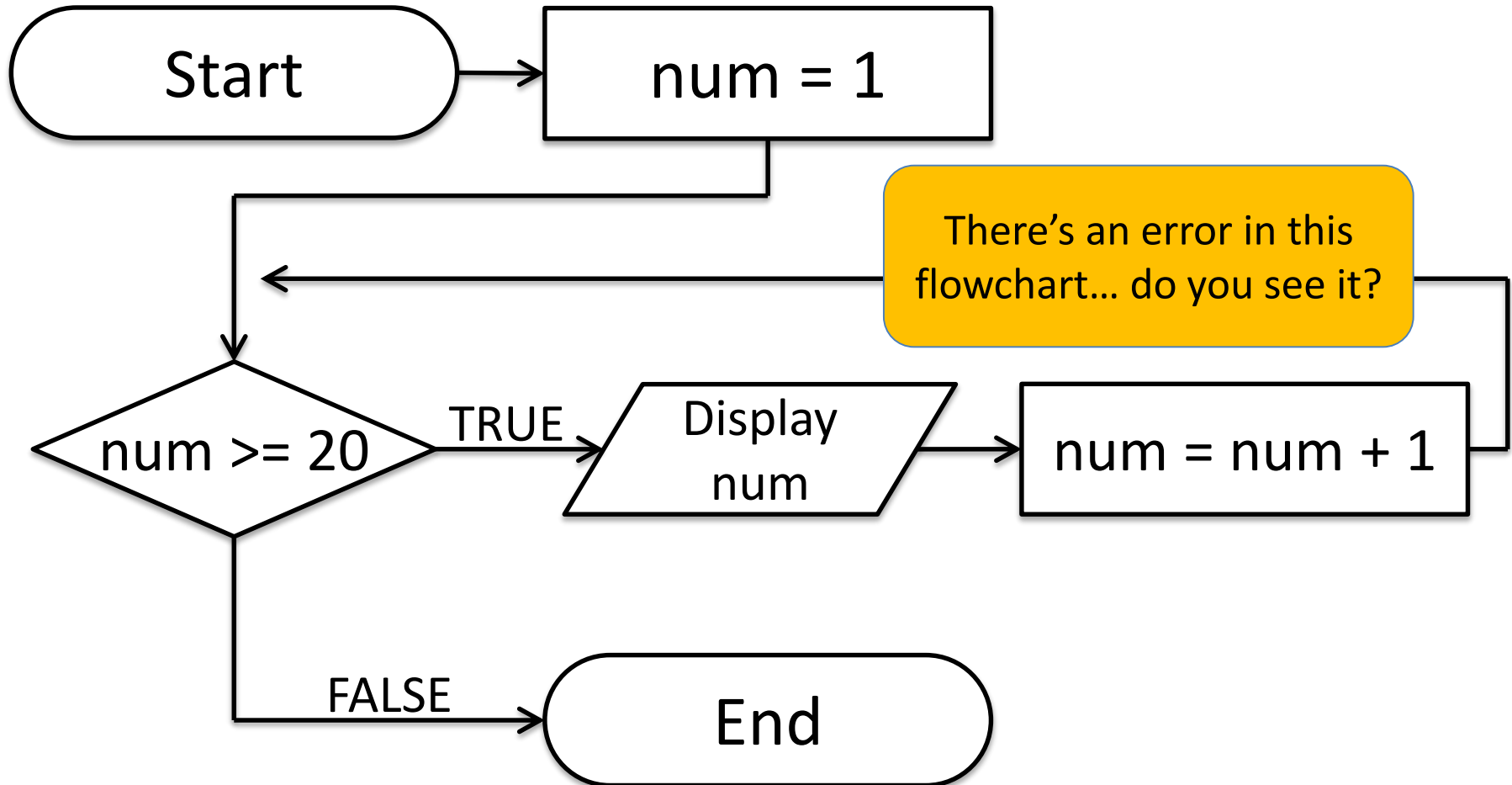
- Doing something over and over (and over) again
- Used in combination with decision making
  - Otherwise we loop forever
    - This is called an “infinite loop”
- What are some real life examples?
  - Doing homework problem sets
  - Walking up steps

# Looping: Pseudocode

- Write an algorithm that counts from 1 to 20
  - Start with a plain English description

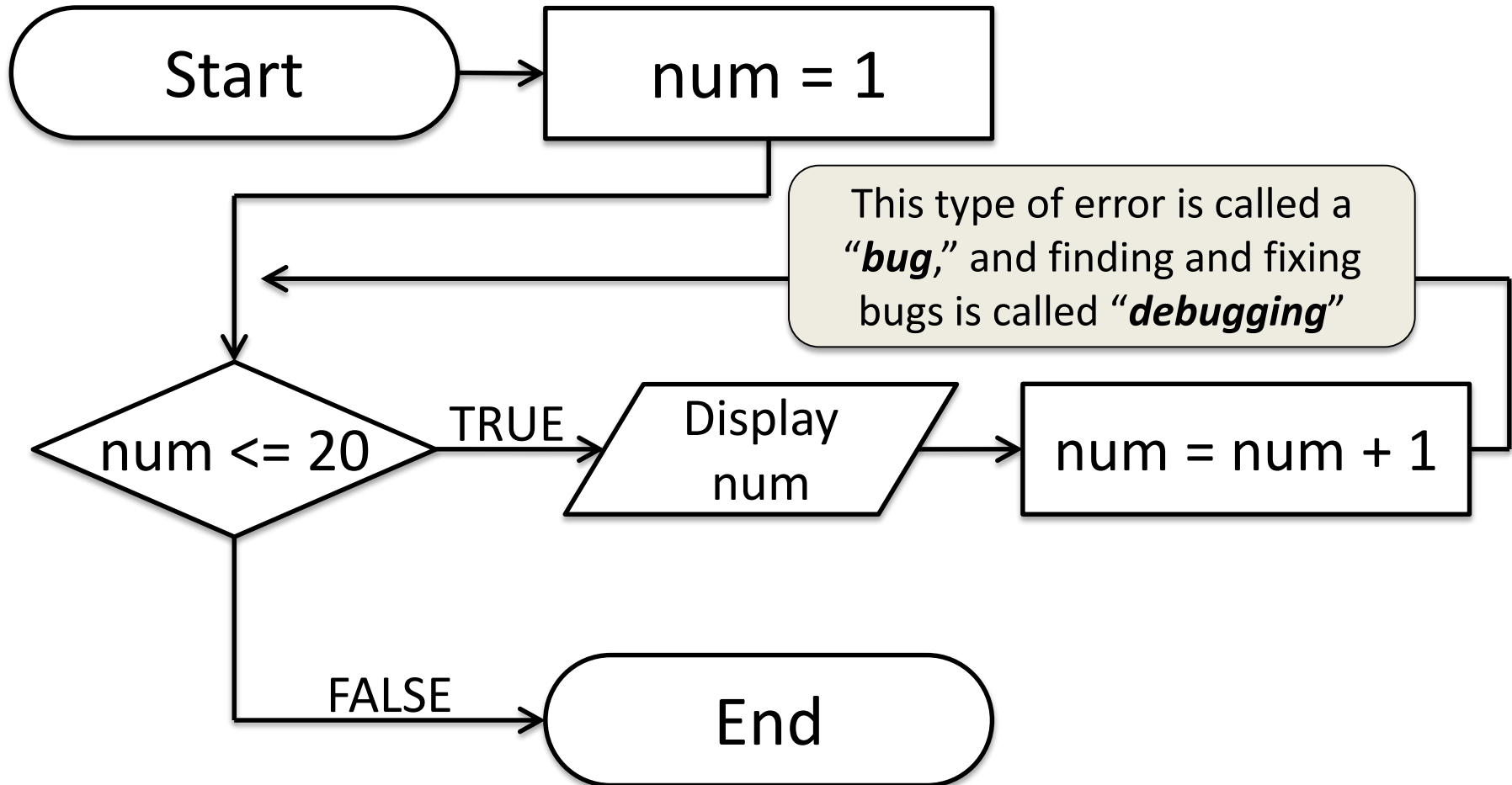
```
1. Set num = 1
2. While num <= 20
3.     Display num
4.     num = num + 1
5. (End loop)
```

# Looping: Flowchart





# Looping: Flowchart



# Debugging

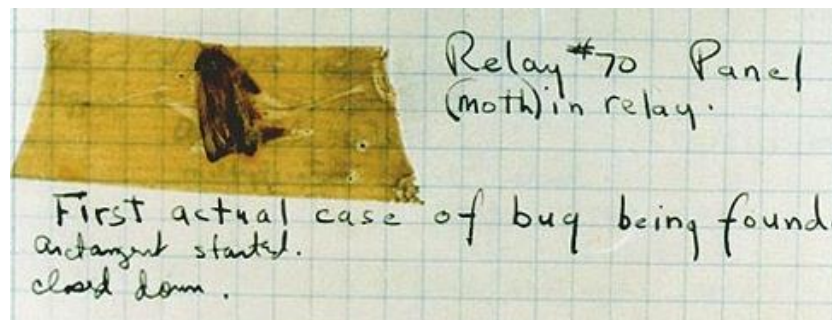
# A Bit of History on “Bugs”



Rear Admiral Grace Hopper

- US Navy lab – September 9, 1947
- Grace Hopper and her colleagues were working on the Harvard Mark II – Or trying to... it wasn't working right

- They found a literal bug inside the machine
  - Taped the bug (a moth) 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

# 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:
    - ("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

# 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
    - “Close jelly jar. Put jelly on bread. Open jelly jar.”



# Exercise

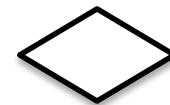
- Write an algorithm that asks a user for their name, then responds with “Hello NAME”
- You can use a flowchart or pseudocode



Input/Output



Data Processing



Decision



Flow Control

## Exercise #2

- Write an algorithm that asks a user for their grade, and tells them their letter grade.

A: 100 - 90

C: <80 - 70

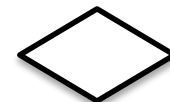
F: <60 - 0

B: <90 - 80

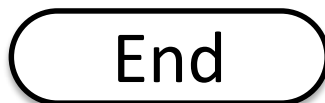
D: <70 - 60



Input/Output



Decision



Data Processing



Flow Control

# Announcements

- Lab 1 is an online lab this week
  - Due by Thursday @ 8:59:59 PM
- Labs will meet in person starting next week
- Homework 1 will be out Wednesday night
  - Due Wednesday (September 14<sup>th</sup>) at 8:59:59 PM
  - You must complete the “Syllabus and Course Website Quiz” before being able to access it
  - Both of these assignments are on Blackboard