CMSC201
Computer Science I for Majors

Lecture 02 – Algorithmic Thinking

Prof. Katherine Gibson

Last Class We Covered

- Syllabus
  - Grading scheme, expectations, etc.
  - Academic Integrity Policy
- Computer System Components
- 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?

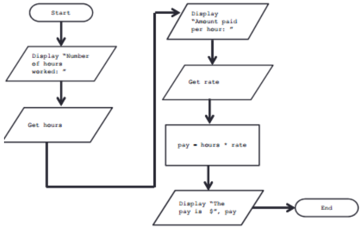
“Weekly Pay” Example

- 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


Start
Display "Number of hours worked: "
Get hours
Display "Amount paid per hour: "
Get rate
pay = hours * rate
Display "The pay is $", pay
End

- Flowchart
  - Symbols convey different types of actions
- Pseudocode
  - A cross between code and plain English

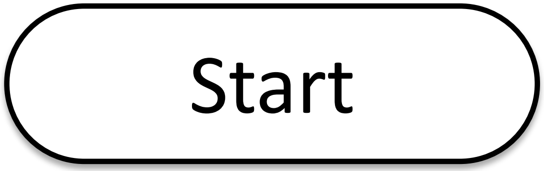
- One may be easier for you – use that one

Step 2A: Pseudocode

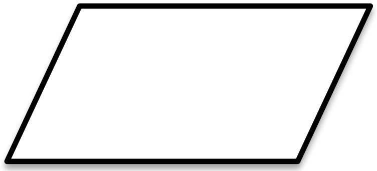
- Start with a plain English description, then…

```
1. Variables: hours, rate, pay
2. Display “Number of hours worked: ”
3. Get hours
4. Display “Amount paid per hour: ”
5. Get rate
6. pay = hours * rate
7. Display “The pay is $” , pay
```

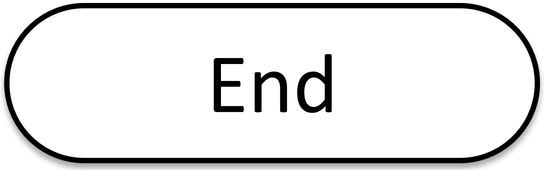
Flowchart Symbols



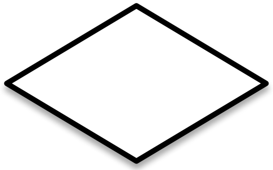
Start Symbol



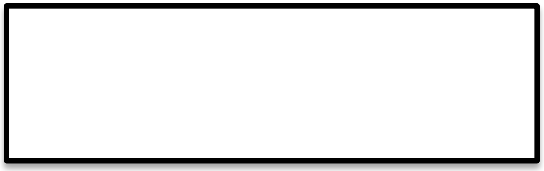
Input/Output



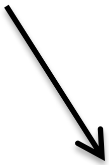
End Symbol



Decision Symbol

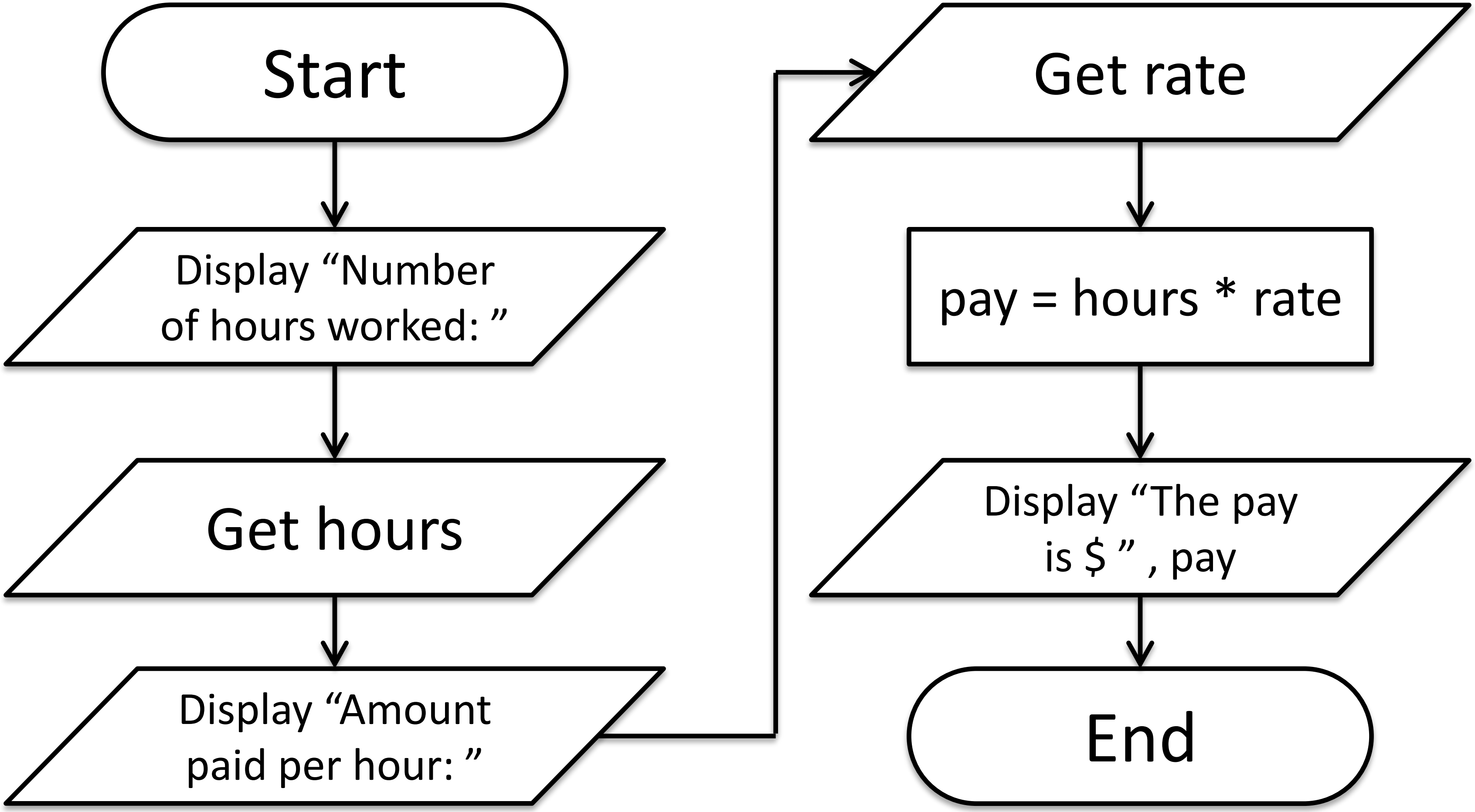


Data Processing Symbol



Flow Control Arrows

Step 2B: Flowchart


Start
Display "Number of hours worked: "
Get hours
Display "Amount paid per hour: "
Get rate
pay = hours * rate
Display "The pay is $ " , pay
End

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 control structures we can use in our algorithms

Control Structures

Control Structures

- Structures that control how the program "flows" or operates, and in which 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***

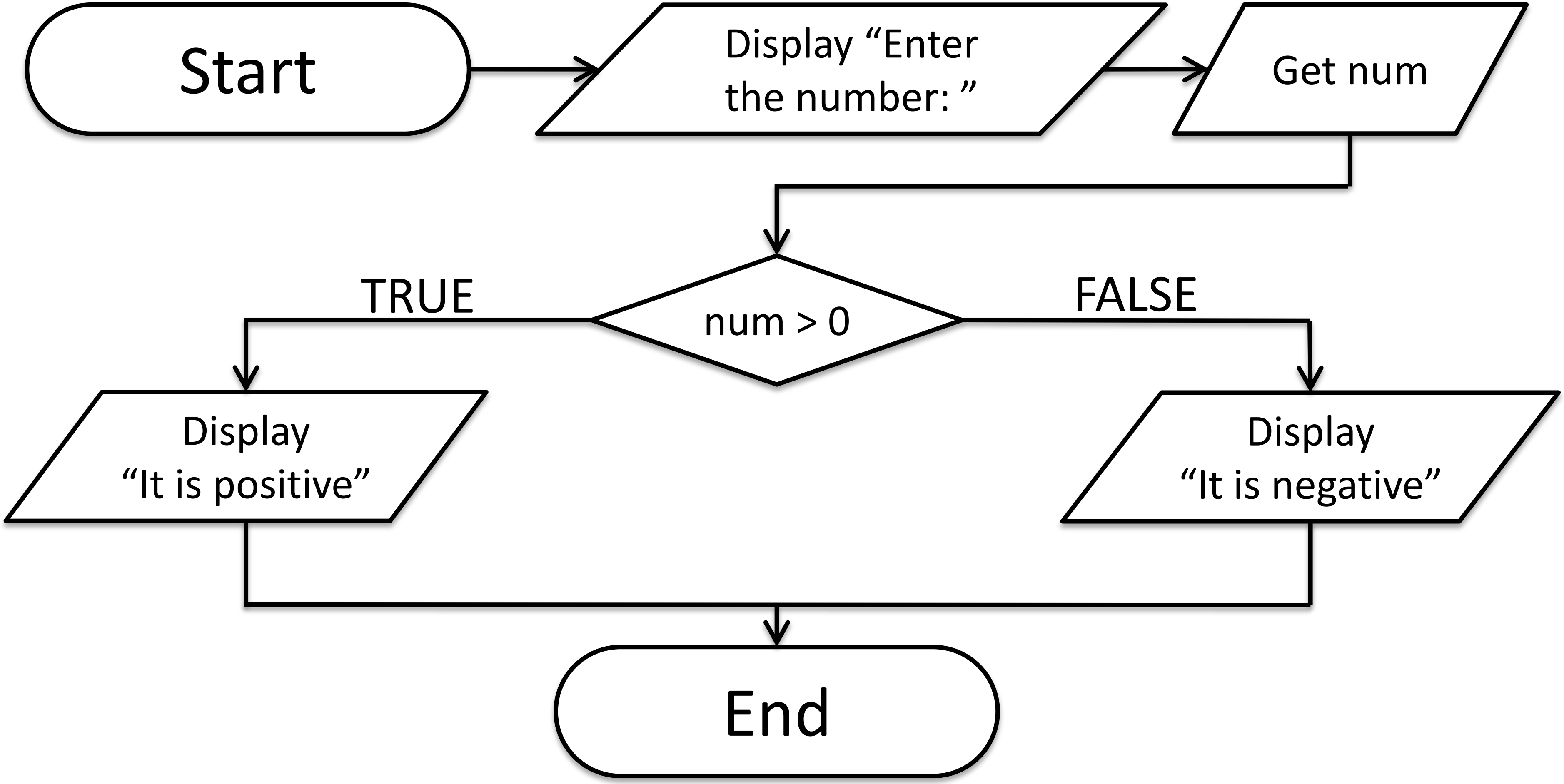
- What are some real life examples?
  - Walking around campus (construction!)
  - Choosing where to eat for lunch

Decision Making: Pseudocode

- Answer the question “Is a number positive?”
  - Start with a plain English description

```
Variable:  num
Display “Enter the number: ”
Get num
If num > 0
     Display “It is positive”
Else
     Display “It is negative”
```

Decision Making: Flowchart



Looping

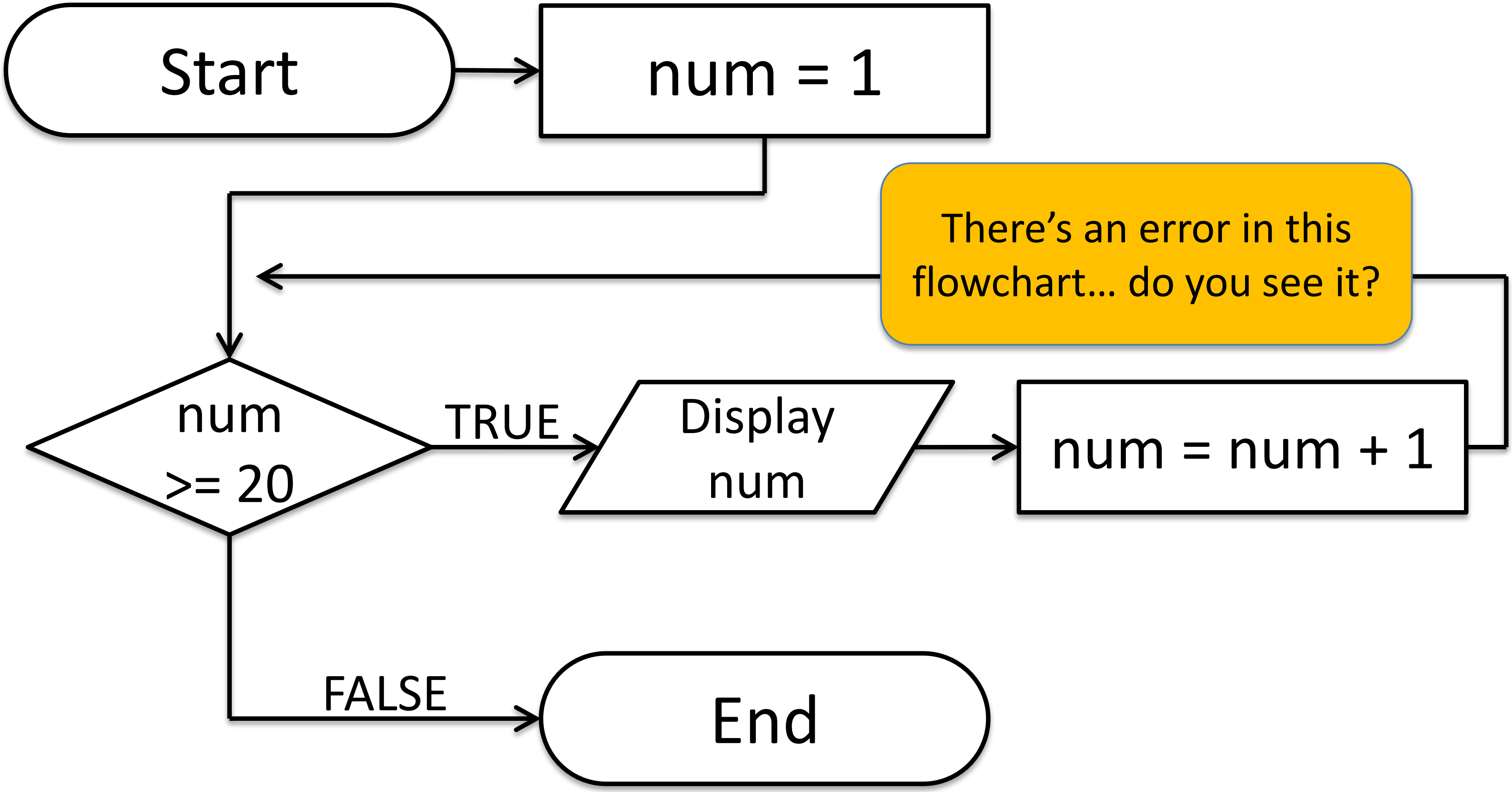
- Doing something over and over again
- Combined with decision making
  - Otherwise we loop forever (an “infinite loop”)
- What are some real life examples?
  - Doing homework problem sets
  - Walking up steps

Looping: Pseudocode

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

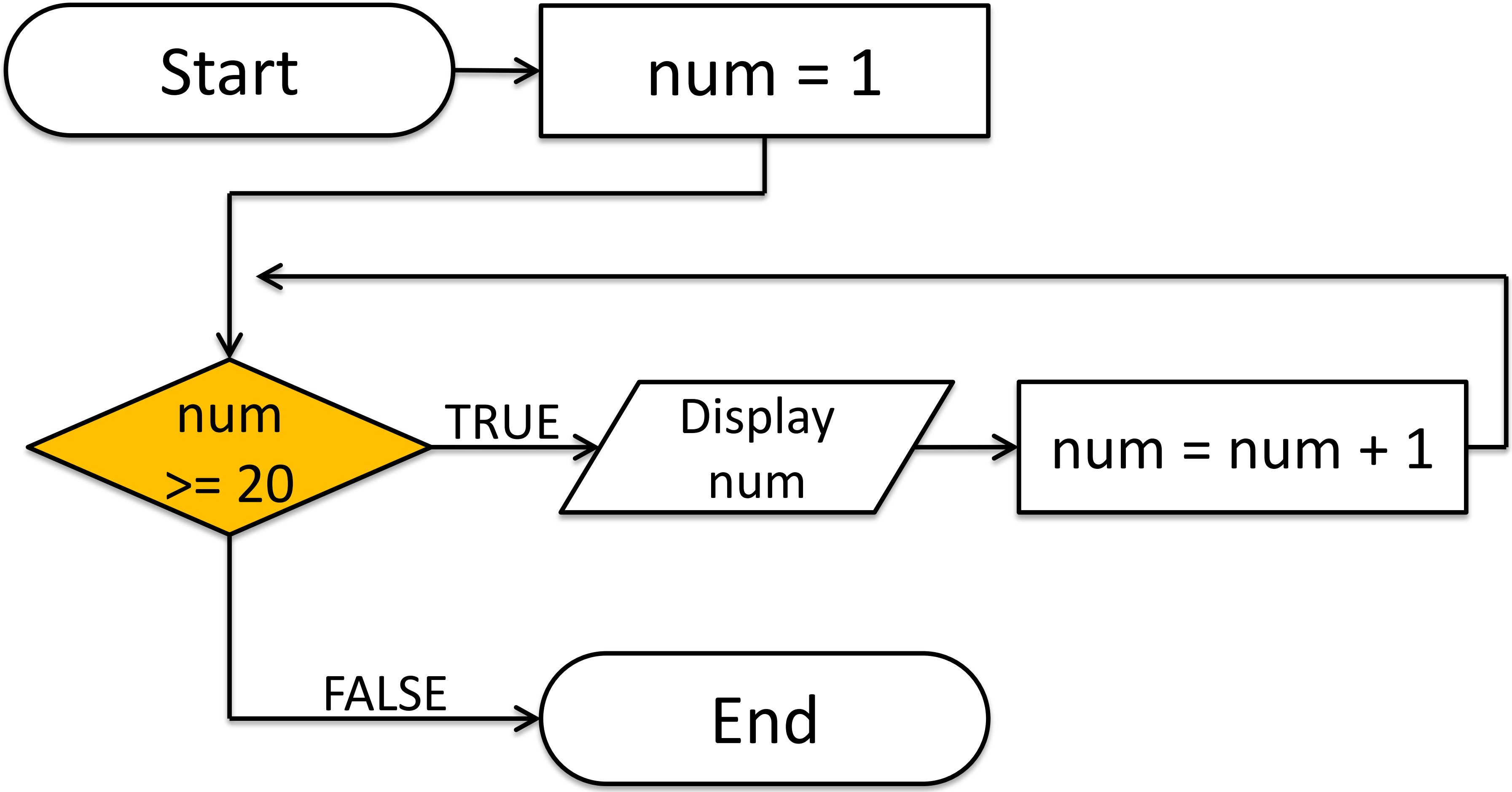
```
Variable: num
num = 1
While num <= 20
    Display num
    num = num + 1
(End loop)
```

Looping: Flowchart

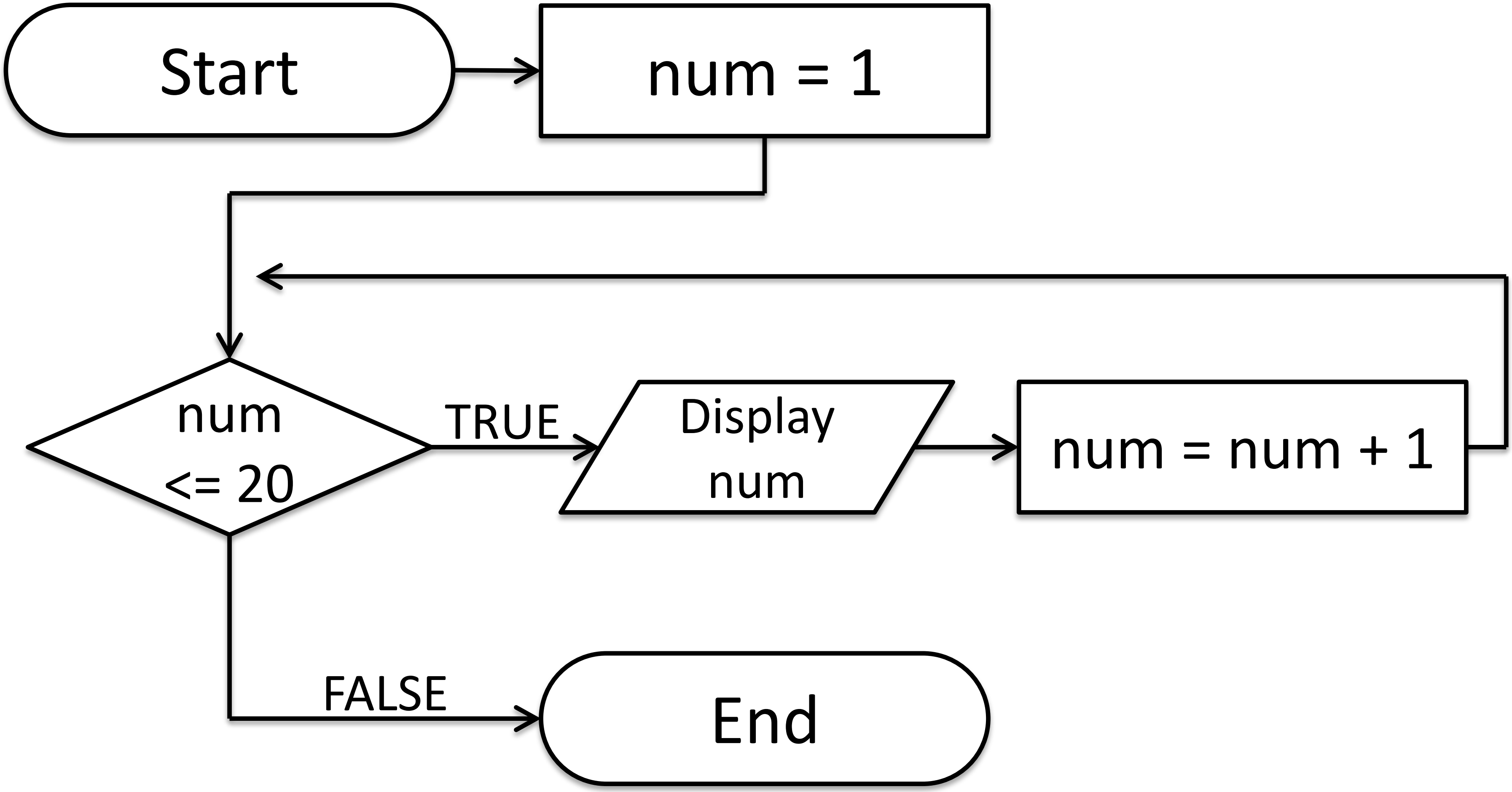


There's an error in this flowchart… do you see it?

Looping: Flowchart



Looping: Flowchart

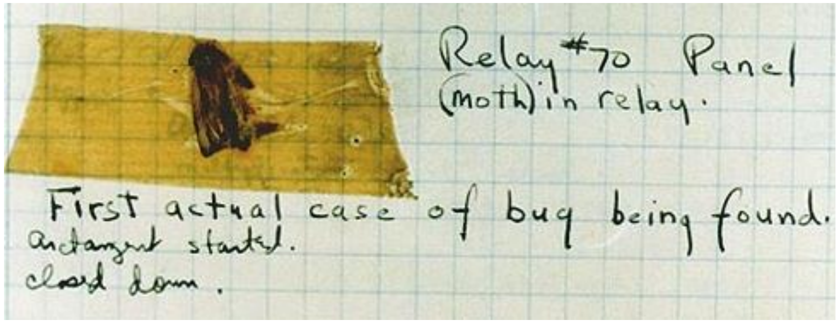


Debugging

A Bit of History on “Bugs”

- US Navy lab – September 9, 1947
- Grace Hopper and colleagues are 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



Relay #70 Panel
(moth) in relay.

First actual case of bug being found.
arctangent started.
closed down.

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 errors in each line of code below:

```
prnit("Hello")
print("What"s up? ")
print("Aloha!)
print("Good Monring")
```

Syntax Error Examples

- Find the errors in each line of code below:

```
prnit("Hello")
print("What"s up? ")
print("Aloha!)
print("Good Monring")
```

not actually a syntax error

Logical Errors

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

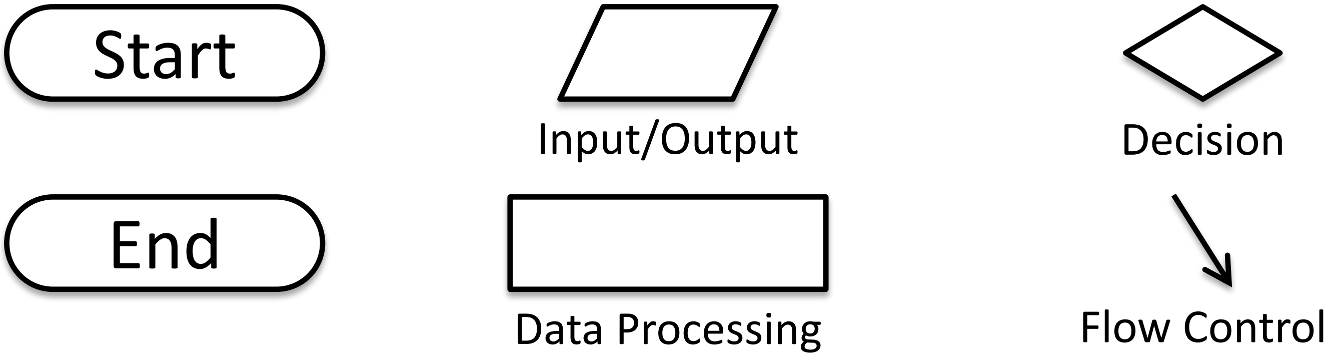
- Examples of logical errors:
  - Using the wrong value for something

    ```
    callMe = “maybe NOT”
    ```

  - Doing steps in the wrong order
    - “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



Start

End

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.

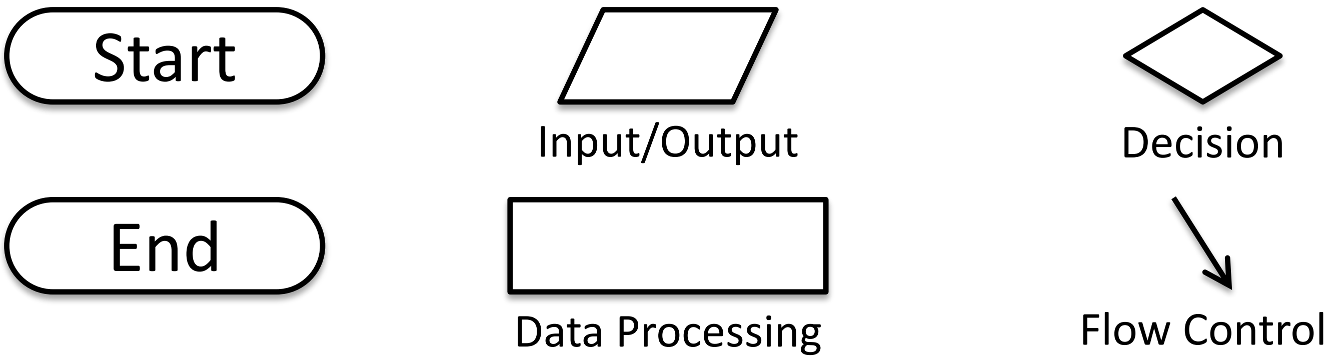
A: 100-90

B: 90-80

C: 80-70

D: 70-60

F: 60-0



Start

End

Input/Output

Data Processing

Decision

Flow Control

Announcements

- Your Lab 1 is an online lab this week!
  - Due by this Thursday (Sept 3rd) at 8:59:59 PM

- Homework 1 is out
  - Due by next Tuesday (Sept 8th) at 8:59:59 PM

- Both of these assignments are on Blackboard
  - Weekly Agendas are also on Blackboard