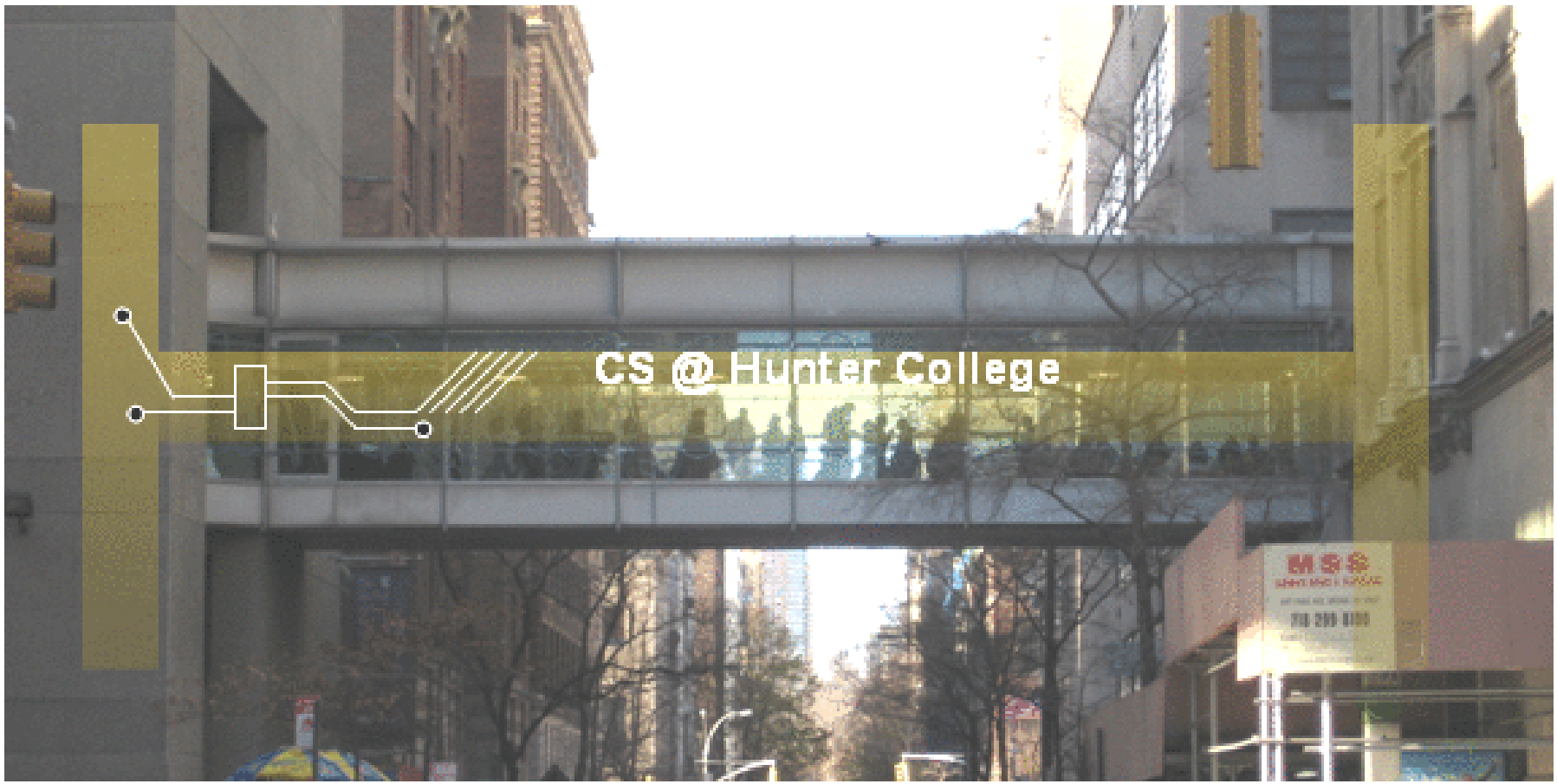
CSci 127: Introduction to Computer Science



Finished the lecture preview?

hunter.cuny.edu/csci

Guest Speakers

- Today we will start with guest speakers from computer science clubs at Hunter
- Instead of the usual lecture slip, you will fill out a survey to get credit for today's lecture (link on Blackboard under Course Materials)

Challenge (Group Work):

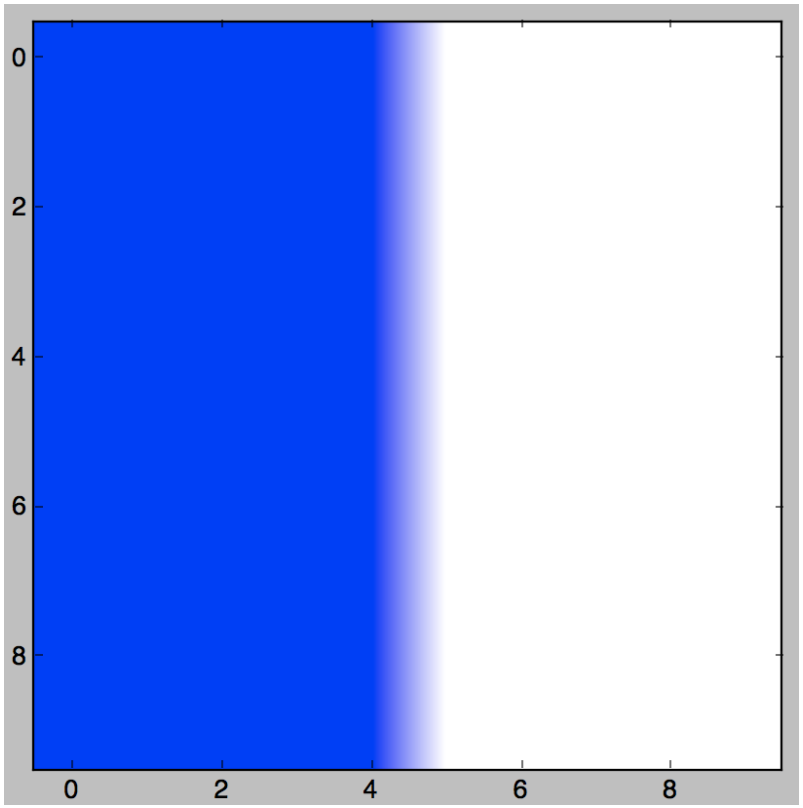
- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.ones( (10,10,3) )
    img[0:10,0:5,0:2] = 0
    ```
    ```

Challenge (Group Work):

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.ones( (10,10,3) )
    img[0:10,0:5,0:2] = 0
    ```


    ```

Challenge (Group Work)

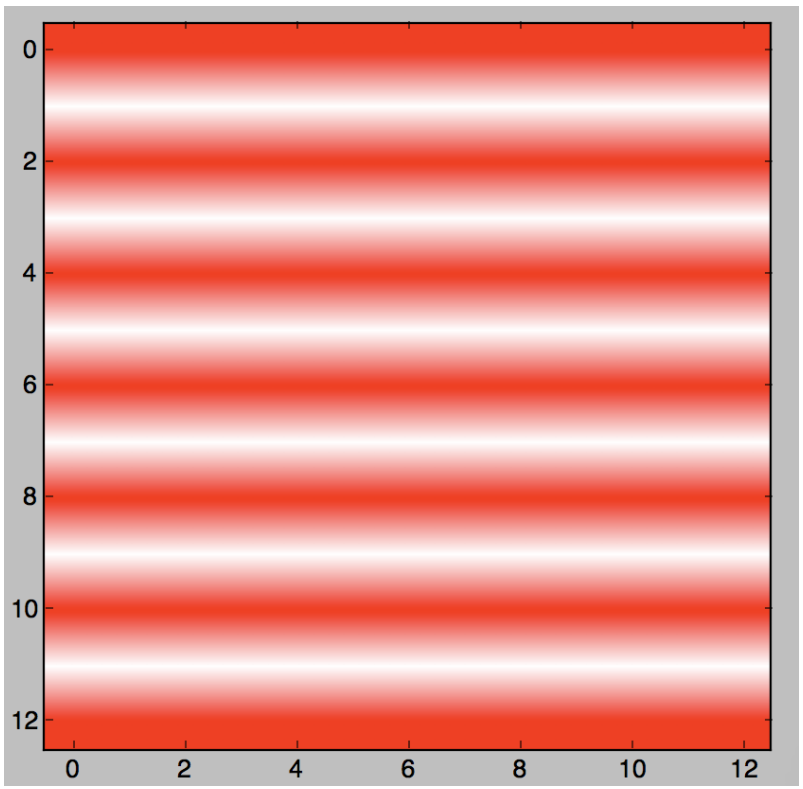
- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    num = int(input('Enter size '))
    img = np.ones( (num,num,3) )
    img[::2,:,1:]  = 0
    ```
    ```

Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    num = int(input('Enter size '))
    img = np.ones( (num,num,3) )
    img[::2,:,1:]  = 0
    ```


    ```

Challenge (Group Work)

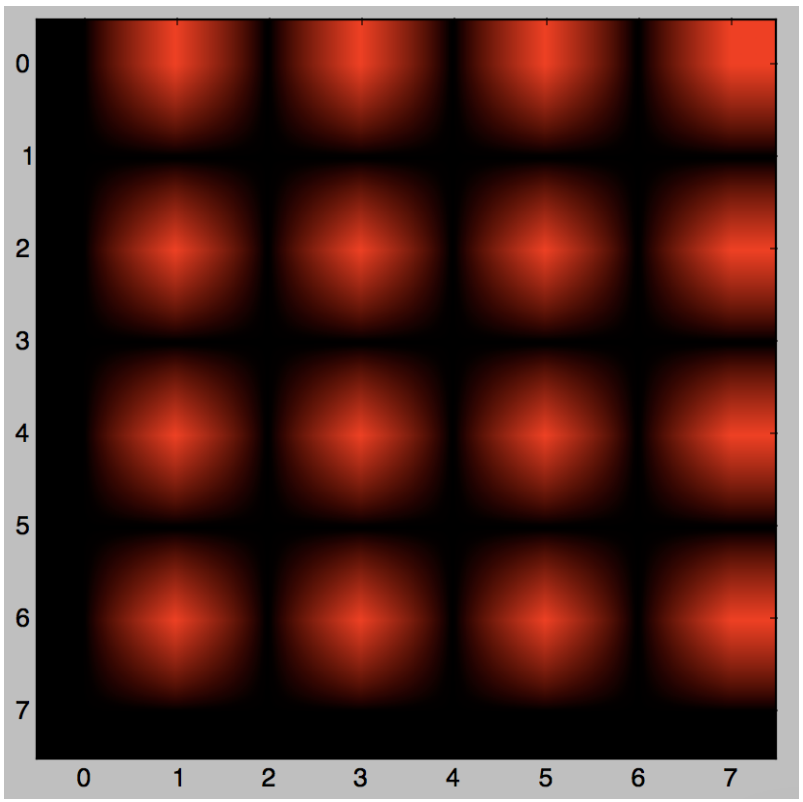
- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.zeros( (8,8,3) )
    img[::2,1::2,0] = 1
    ```
    ```

Challenge (Group Work)

- Basic pattern: *img[rows, columns, channels]* with: *start:stop:step*.
- Assuming the libraries are imported, what do the following code fragments produce:

  - ```
    img = np.zeros( (8,8,3) )
    img[::2,1::2,0] = 1
    ```


    ```

Today's Topics

- Recap: Decisions
- Logical Expressions
- Circuits
- Binary Numbers

Today's Topics

- **Recap: Decisions**
- Logical Expressions
- Circuits
- Binary Numbers

Decisions

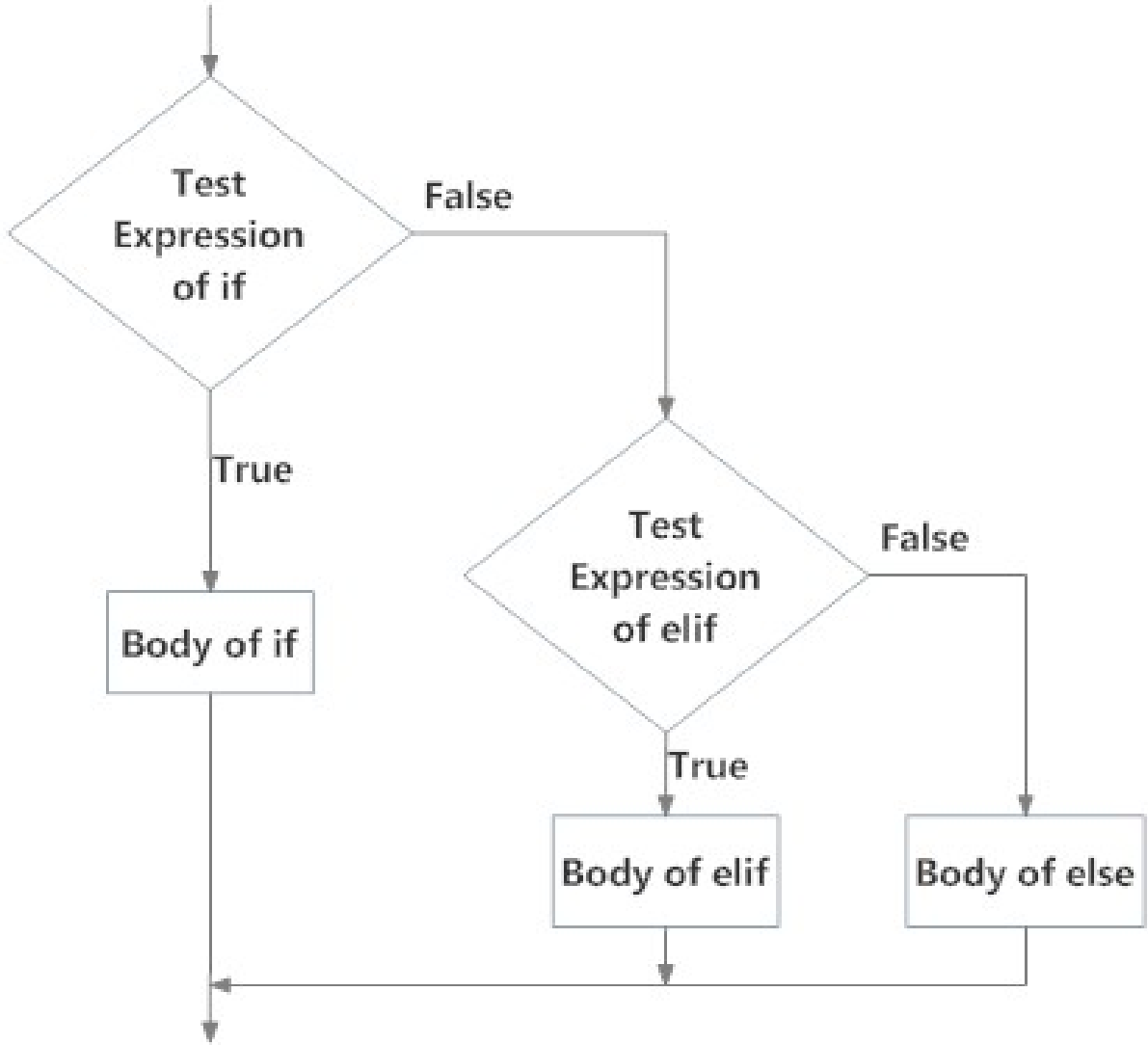
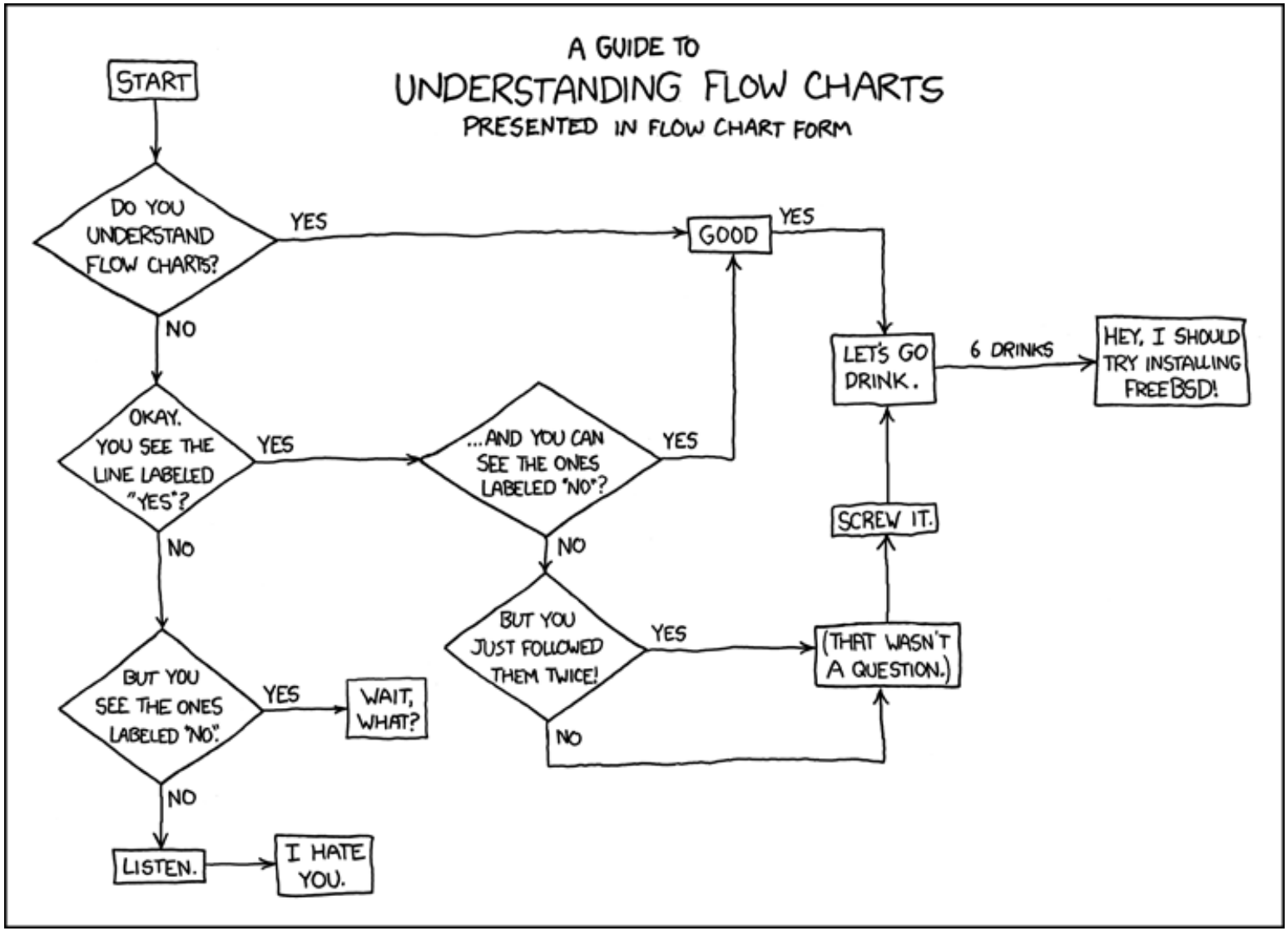


Fig: Operation of if...elif...else statement

(programiz)

Side Note: Reading Flow Charts



(xkcd/518)

Today's Topics

- Recap: Decisions
- **Logical Expressions**
- Circuits
- Binary Numbers

Logical Operators

and

in1		in2	*returns:*
`False`	`and`	`False`	`False`
`False`	`and`	`True`	`False`
`True`	`and`	`False`	`False`
`True`	`and`	`True`	`True`

Logical Operators

and

in1		in2	*returns:*
`False`	`and`	`False`	`False`
`False`	`and`	`True`	`False`
`True`	`and`	`False`	`False`
`True`	`and`	`True`	`True`

or

in1		in2	*returns:*
`False`	`or`	`False`	`False`
`False`	`or`	`True`	`True`
`True`	`or`	`False`	`True`
`True`	`or`	`True`	`True`

Logical Operators

and

in1		in2	*returns:*
`False`	`and`	`False`	`False`
`False`	`and`	`True`	`False`
`True`	`and`	`False`	`False`
`True`	`and`	`True`	`True`

or

in1		in2	*returns:*
`False`	`or`	`False`	`False`
`False`	`or`	`True`	`True`
`True`	`or`	`False`	`True`
`True`	`or`	`True`	`True`

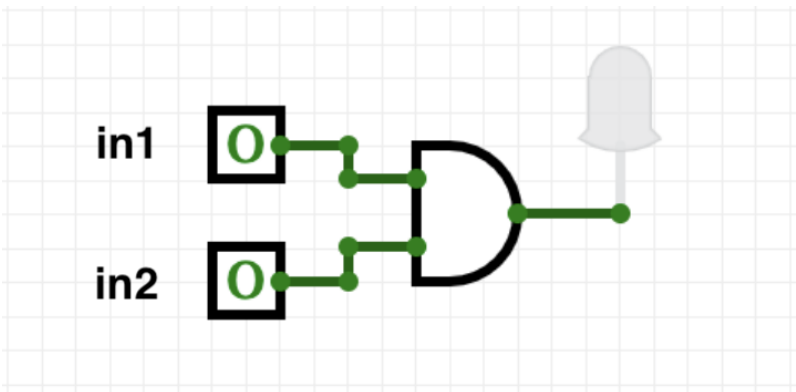
not

	in1	*returns:*
`not`	`False`	`True`
`not`	`True`	`False`

Today's Topics

- Recap: Decisions
- Logical Expressions
- **Circuits**
- Binary Numbers

Circuit Demo

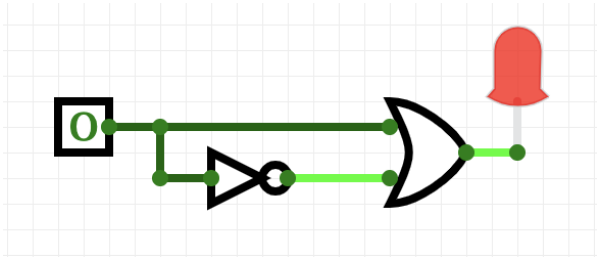


(Demo with `circuitverse`)

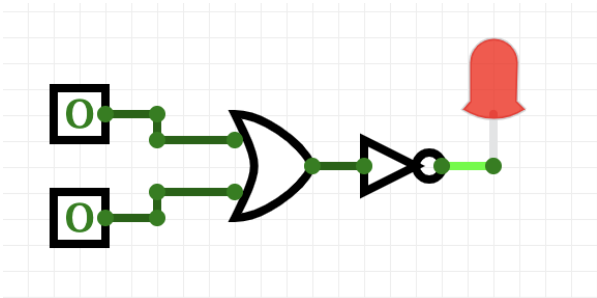
Challenge

Predict when these expressions are true:

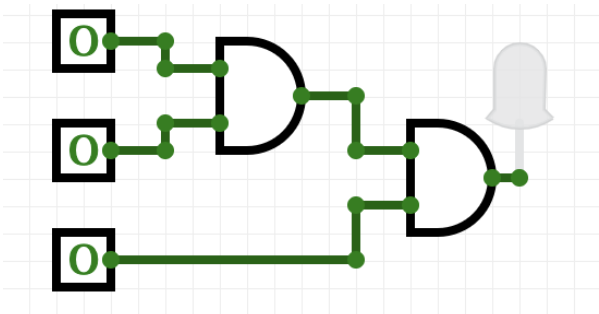
- `in1 or not in1`:



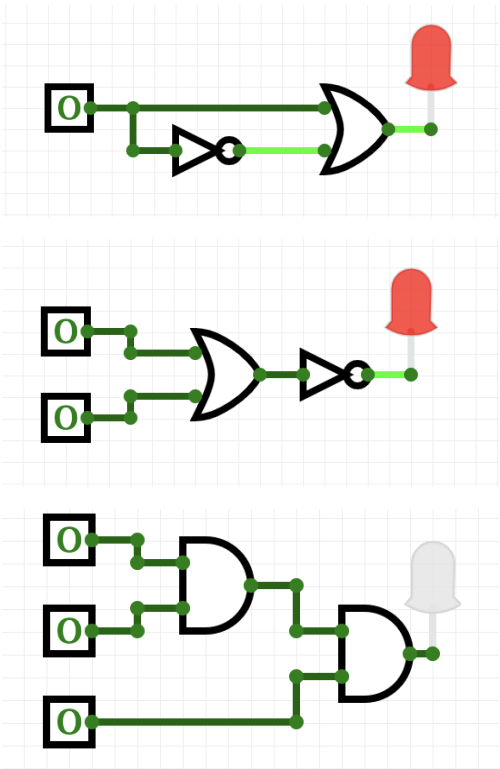
- `not(in1 or in2)`:



- `(in1 and in2) and in3`:

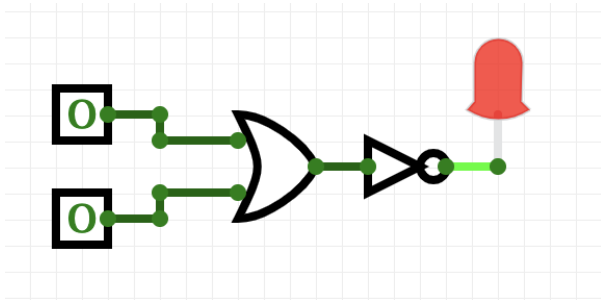


Circuit Demo



(Demo with `circuitverse`)

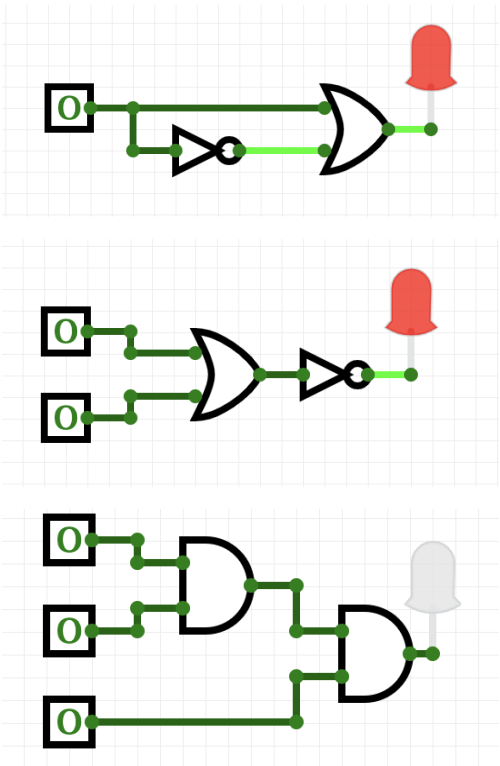
Challenge



Draw a circuit that corresponds to each logical expression:

- `in1 or in2`
- `(in1 or in2) and (in1 or in3)`
- `(not(in1 and not in2)) or (in1 and (in2 and in3))`

Circuit Demo



(Demo with `circuitverse`)

Today's Topics

- Recap: Decisions
- Logical Expressions
- Circuits
- **Binary Numbers**

Binary Numbers

- Logic $\rightarrow$ Circuits $\rightarrow$ Numbers

Binary Numbers

- Logic $\rightarrow$ Circuits $\rightarrow$ Numbers
- Digital logic design allows for two states:

Binary Numbers

- Logic $\rightarrow$ Circuits $\rightarrow$ Numbers
- Digital logic design allows for two states:
  - True / False

Binary Numbers

- Logic $\rightarrow$ Circuits $\rightarrow$ Numbers
- Digital logic design allows for two states:
  - True / False
  - On / Off (two voltage levels)

Binary Numbers

- Logic $\rightarrow$ Circuits $\rightarrow$ Numbers
- Digital logic design allows for two states:
  - True / False
  - On / Off (two voltage levels)
  - 1 / 0

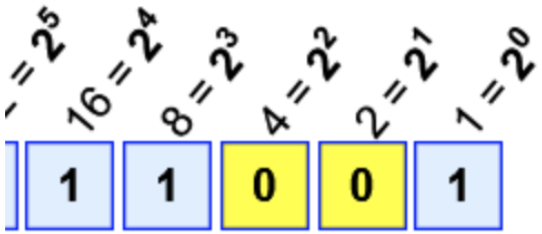
Binary Numbers

- Logic $\rightarrow$ Circuits $\rightarrow$ Numbers
- Digital logic design allows for two states:
  - True / False
  - On / Off (two voltage levels)
  - 1 / 0
- Computers store numbers using the Binary system (base 2)

Binary Numbers

- Logic $\rightarrow$ Circuits $\rightarrow$ Numbers
- Digital logic design allows for two states:
  - True / False
  - On / Off (two voltage levels)
  - 1 / 0
- Computers store numbers using the Binary system (base 2)
- A **bit** (binary digit) being 1 (on) or 0 (off)

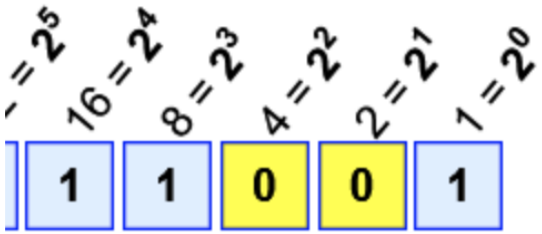
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**

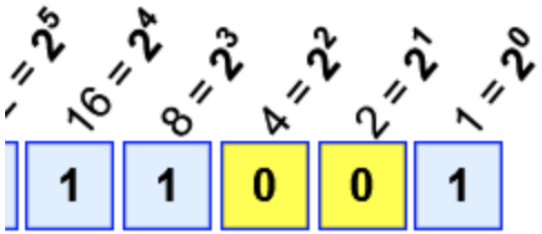
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two

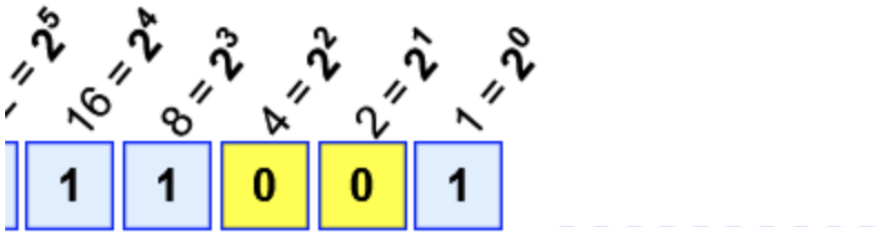
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two
  - Decimal: the "ones", "tens", "hundreds" and so on (powers of 10)

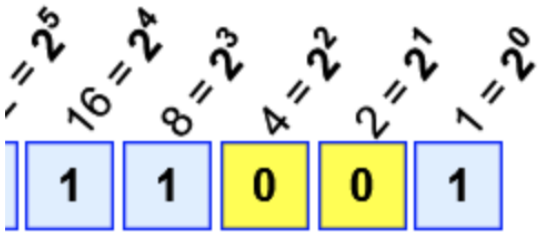
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two
  - Decimal: the "ones", "tens", "hundreds" and so on (powers of 10)
  - Binary: the "ones", "twos", "fours", "sixteens" and so on (powers of 2)

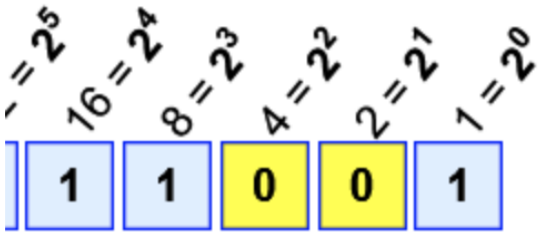
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two
  - Decimal: the "ones", "tens", "hundreds" and so on (powers of 10)
  - Binary: the "ones", "twos", "fours", "sixteens" and so on (powers of 2)
- In each position the digit is either 0 or 1, so given a binary number we can obtain the decimal equivalent as follows:

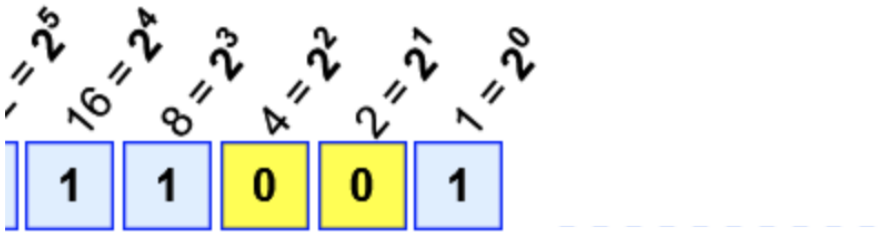
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two
  - Decimal: the "ones", "tens", "hundreds" and so on (powers of 10)
  - Binary: the "ones", "twos", "fours", "sixteens" and so on (powers of 2)
- In each position the digit is either 0 or 1, so given a binary number we can obtain the decimal equivalent as follows:
  - In the "ones" position we either have a 1 or not

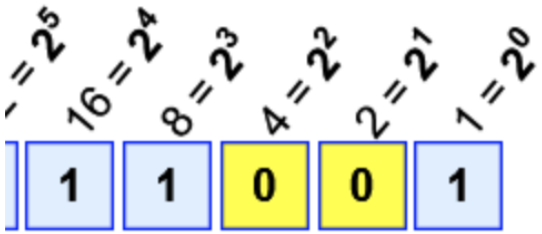
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two
  - Decimal: the "ones", "tens", "hundreds" and so on (powers of 10)
  - Binary: the "ones", "twos", "fours", "sixteens" and so on (powers of 2)
- In each position the digit is either 0 or 1, so given a binary number we can obtain the decimal equivalent as follows:
  - In the "ones" position we either have a 1 or not
  - In the "twos" position we either have a 2 or not

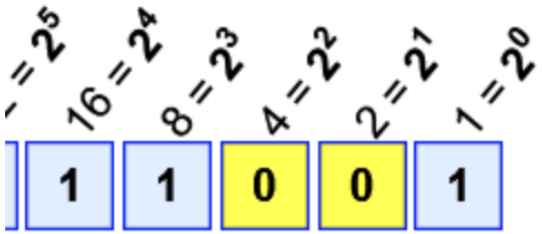
Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two
  - Decimal: the "ones", "tens", "hundreds" and so on (powers of 10)
  - Binary: the "ones", "twos", "fours", "sixteens" and so on (powers of 2)
- In each position the digit is either 0 or 1, so given a binary number we can obtain the decimal equivalent as follows:
  - In the "ones" position we either have a 1 or not
  - In the "twos" position we either have a 2 or not
  - In the "fours" position we either have a 4 or not ...

Binary Numbers



Example: 1×16 + 1×8 + 1×1 = 16+8+1 = 25

- Two digits: **0** and **1**
- Each position is a power of two
  - Decimal: the "ones", "tens", "hundreds" and so on (powers of 10)
  - Binary: the "ones", "twos", "fours", "sixteens" and so on (powers of 2)
- In each position the digit is either 0 or 1, so given a binary number we can obtain the decimal equivalent as follows:
  - In the "ones" position we either have a 1 or not
  - In the "twos" position we either have a 2 or not
  - In the "fours" position we either have a 4 or not ...
- **Example:**

$$11001_{base2} = 16 + 8 + 1 = 25_{base10}$$

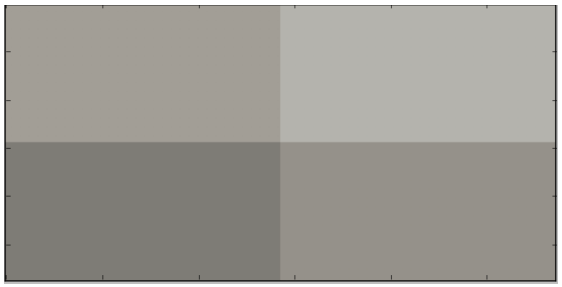
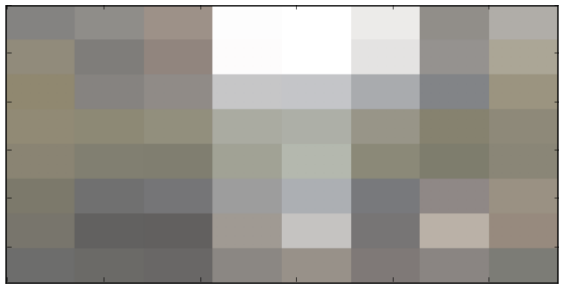
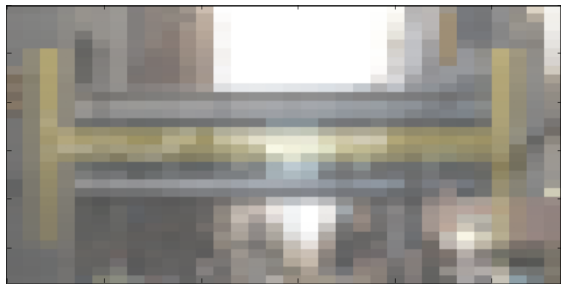
Recap

- In Python, we introduced:

Recap

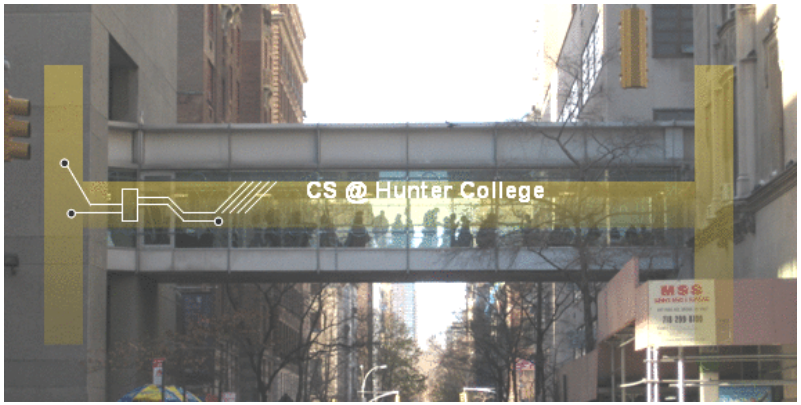
- In Python, we introduced:
  - Decisions
  - Logical Expressions
  - Circuits
  - Binary Numbers

Final Exam

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final Exam Information).

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab

Weekly Reminders!

Before next lecture, don't forget to:

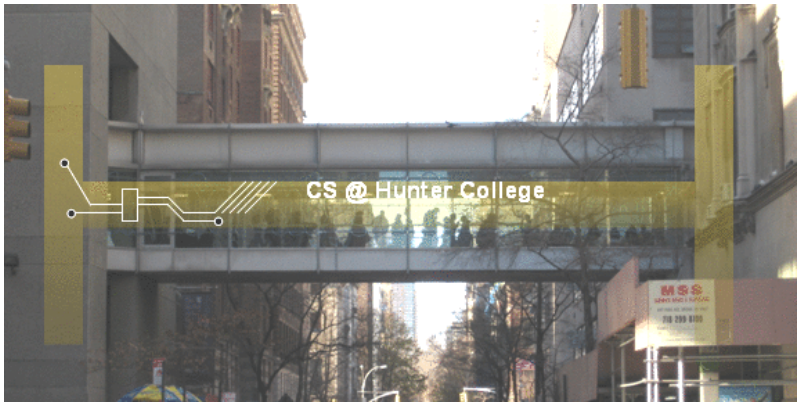
- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North

Weekly Reminders!

Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review in lab 1001G Hunter North

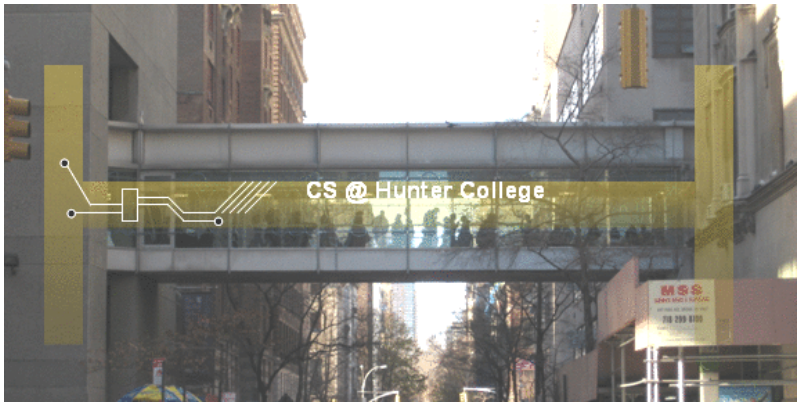
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review in lab 1001G Hunter North
- Submit this week's 5 programming assignments

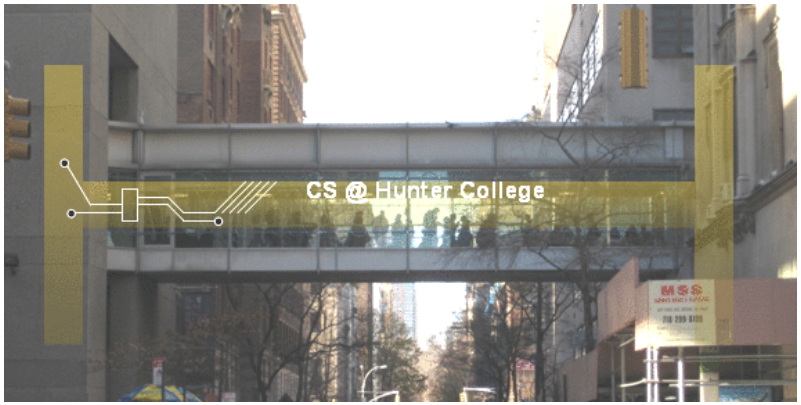
Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review in lab 1001G Hunter North
- Submit this week's 5 programming assignments
- If you need help, schedule an appointment for Tutoring in lab 1001G 11am-5pm

Weekly Reminders!



Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review in lab 1001G Hunter North
- Submit this week's 5 programming assignments
- If you need help, schedule an appointment for Tutoring in lab 1001G 11am-5pm
- Take the Lecture Preview on Blackboard on Monday (or no later than 10am on Tuesday)

Lecture Slips & Writing Boards

- Hand your lecture slip to a UTA.
- Return writing boards as you leave.