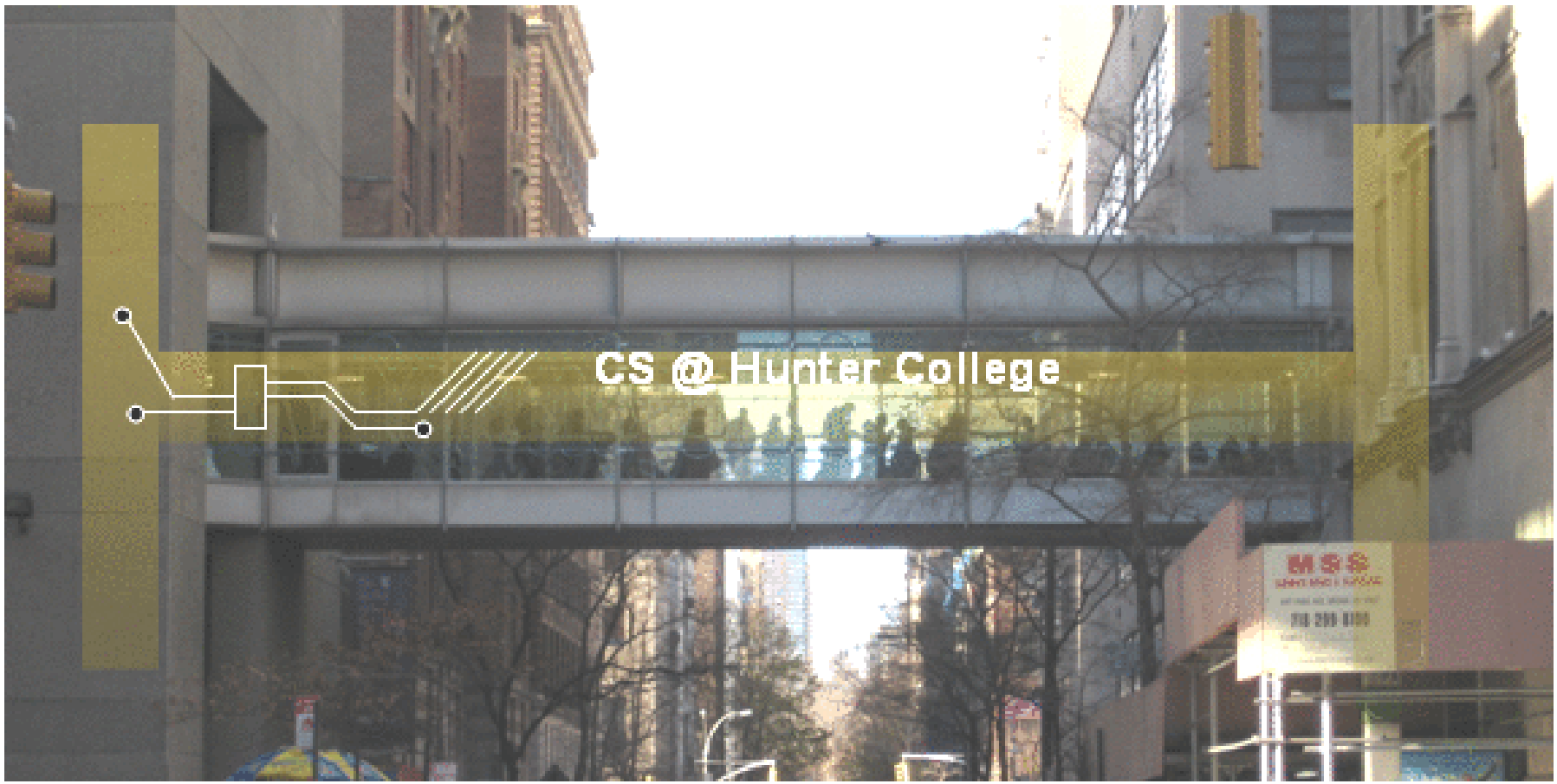
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*
- **How can I tell strings from variables?**

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*
- **How can I tell strings from variables?**
  Strings are surrounded by quotes (either single or double).

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*
- **How can I tell strings from variables?**
  Strings are surrounded by quotes (either single or double).
  Variables names (identifiers) for memory locations are not.

Frequently Asked Questions

From lecture slips & recitation sections.

- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*
- **How can I tell strings from variables?**
  Strings are surrounded by quotes (either single or double).
  Variables names (identifiers) for memory locations are not. Ex: `'num'` *vs.* `num`.
- **This course is Hybrid, what does that mean?**

Frequently Asked Questions

From lecture slips & recitation sections.

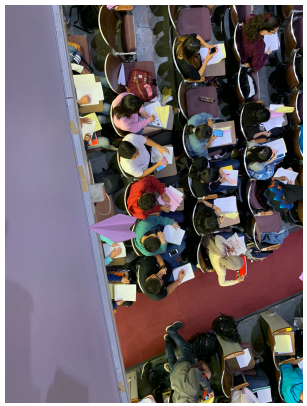
- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*
- **How can I tell strings from variables?**
  Strings are surrounded by quotes (either single or double).
  Variables names (identifiers) for memory locations are not. Ex: `'num'` *vs.* `num`.
- **This course is Hybrid, what does that mean?**
  It means that, instead of a recitation section, each week you independently read the online lab (set aside about 1 hour).

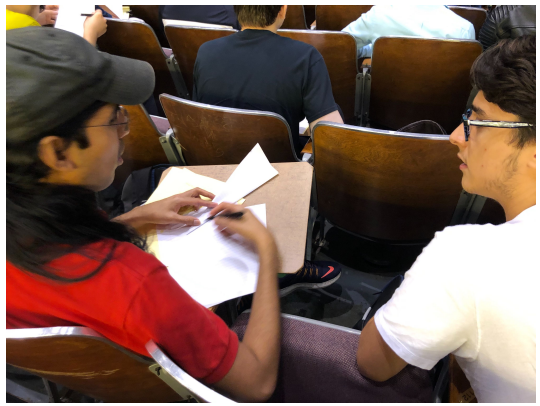
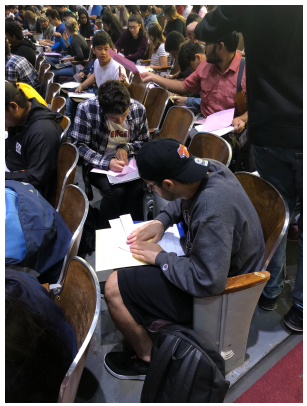
Frequently Asked Questions

From lecture slips & recitation sections.

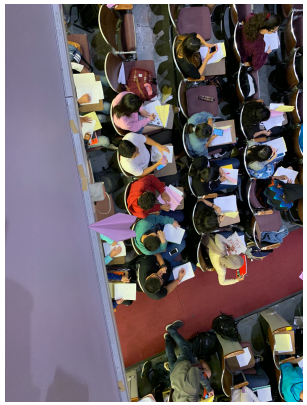
- **Can we do more on colors, images, numpy & matplotlib?**
  Yes, we will in Labs 6-9 & Lectures 6-9.
  Today, we'll focus on decisions, and logical expressions & circuits.
- **What is pseudocode? Why do we use it?**
  Pseudocode is the "informal high-level description of the operating principle of a computer program or other algorithm."
  We use it to write down the ideas, before getting deep into the details.
- **What are types of variables?**
  Different kinds of information takes different amounts of space.
  Types we have seen so far: `int`, `float`, `str` *and objects (e.g. turtles).*
- **How can I tell strings from variables?**
  Strings are surrounded by quotes (either single or double).
  Variables names (identifiers) for memory locations are not. Ex: `'num'` *vs.* `num`.
- **This course is Hybrid, what does that mean?**
  It means that, instead of a recitation section, each week you independently read the online lab (set aside about 1 hour). If you are not reading these thoroughly, you are going to miss a lot of important information (e.g. shell commands)

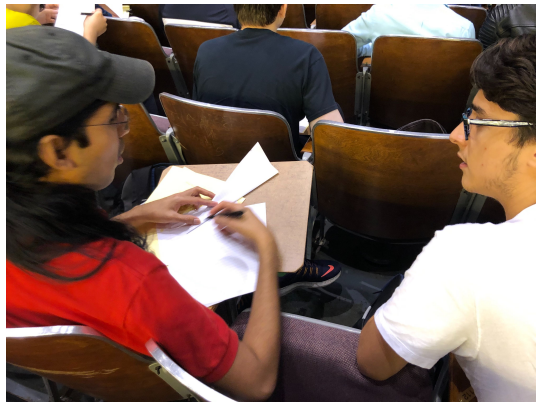
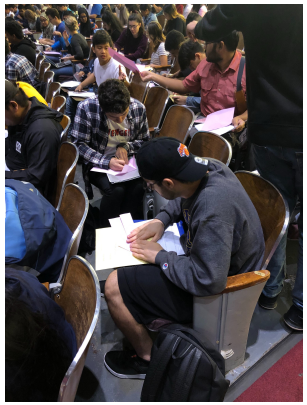
One More FAQ: Why Paper Planes?



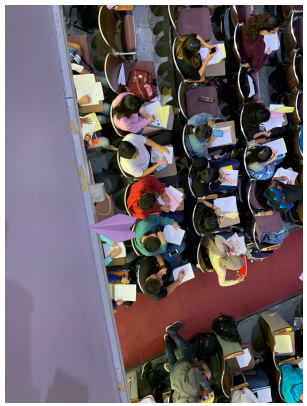
One More FAQ: Why Paper Planes?

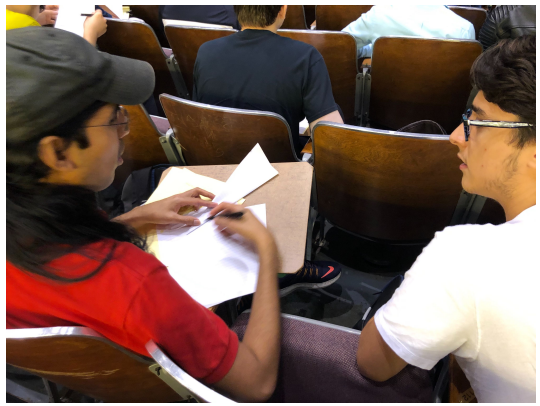
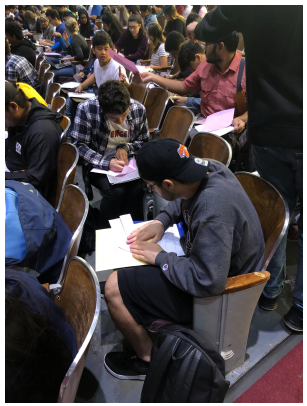


Why paper planes?

One More FAQ: Why Paper Planes?

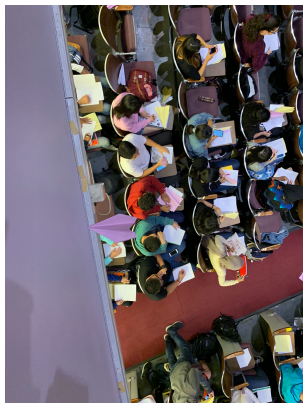


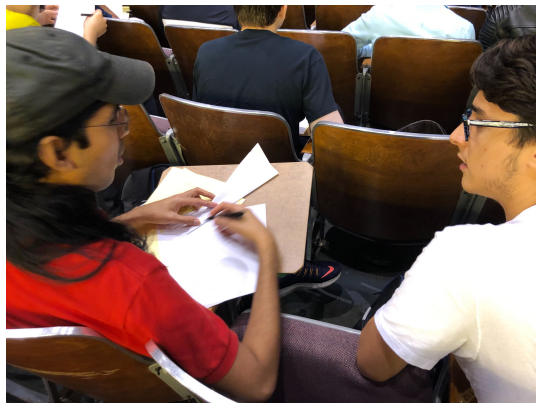
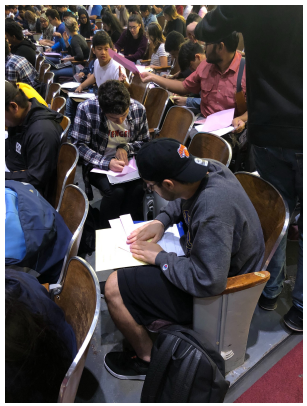



Why paper planes?

- It's a classic design question in introductory programming classes, since

One More FAQ: Why Paper Planes?

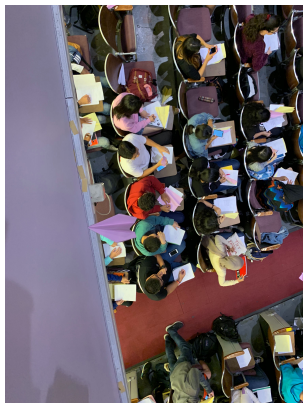


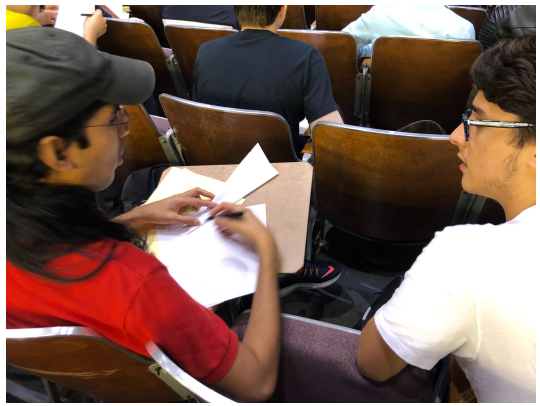
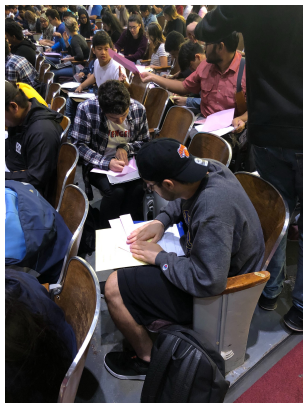



Why paper planes?

- It's a classic design question in introductory programming classes, since
  - Practice writing solutions (algorithms) in plain English without worrying about syntax (pseudocode).

One More FAQ: Why Paper Planes?

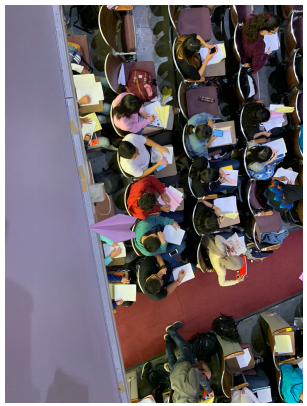


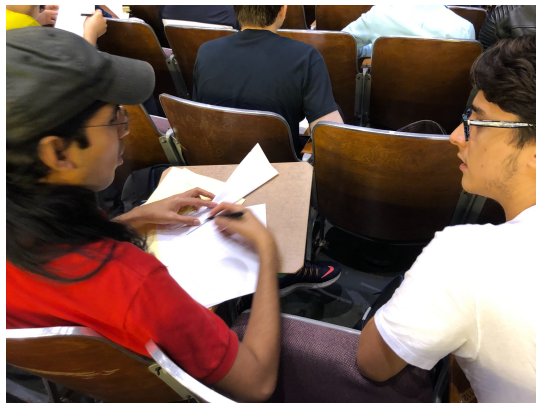
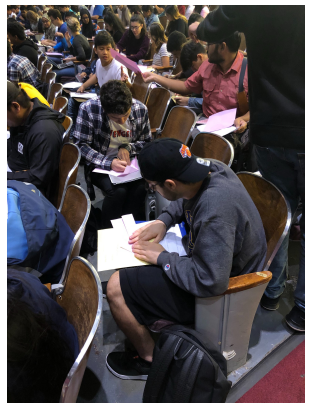



Why paper planes?

- It's a classic design question in introductory programming classes, since
  - Practice writing solutions (algorithms) in plain English without worrying about syntax (pseudocode).
  - Practice thinking (and writing) precisely.

One More FAQ: Why Paper Planes?

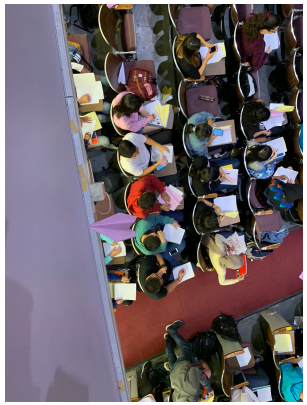


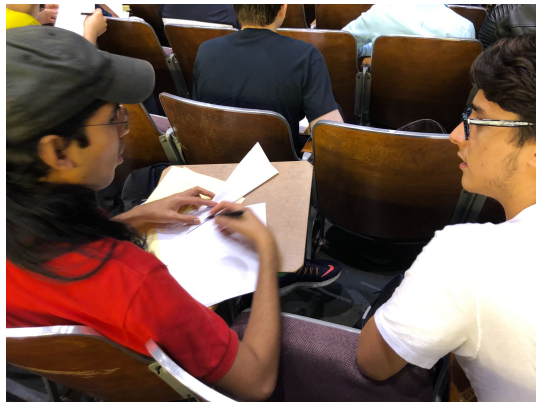
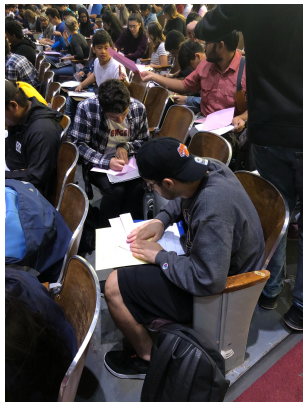



Why paper planes?

- It's a classic design question in introductory programming classes, since
  - Practice writing solutions (algorithms) in plain English without worrying about syntax (pseudocode).
  - Practice thinking (and writing) precisely.
- Why in groups?

One More FAQ: Why Paper Planes?

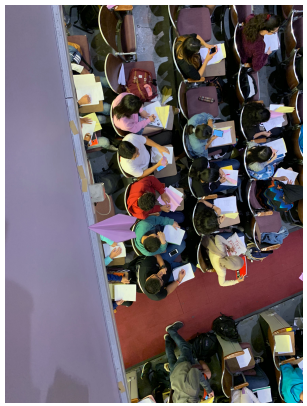


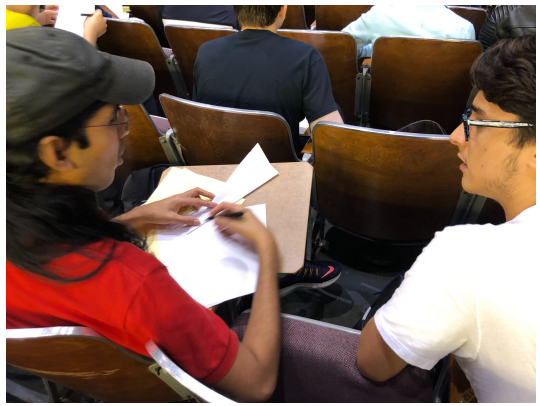
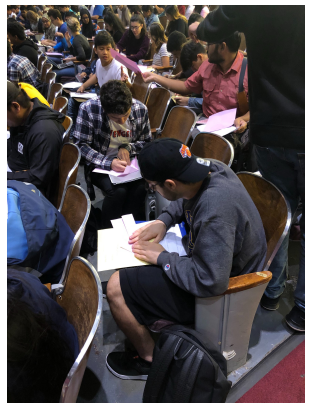



Why paper planes?

- It's a classic design question in introductory programming classes, since
  - Practice writing solutions (algorithms) in plain English without worrying about syntax (pseudocode).
  - Practice thinking (and writing) precisely.
- Why in groups?
  - Improves mastery of material.

One More FAQ: Why Paper Planes?

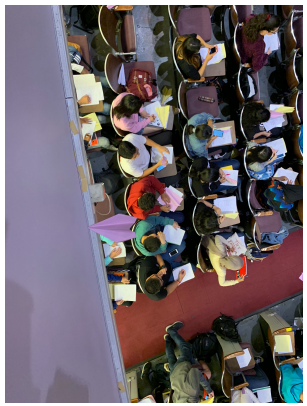


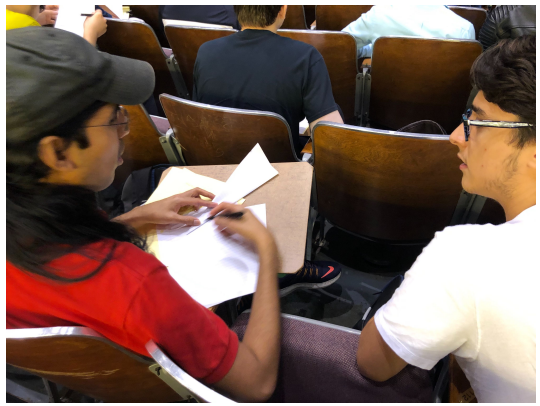
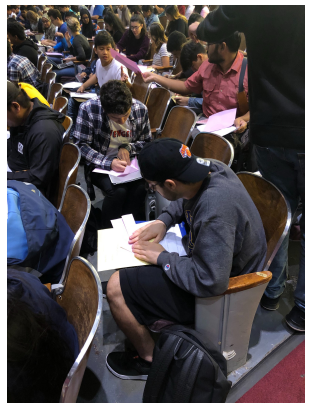



Why paper planes?

- It's a classic design question in introductory programming classes, since
  - Practice writing solutions (algorithms) in plain English without worrying about syntax (pseudocode).
  - Practice thinking (and writing) precisely.
- Why in groups?
  - Improves mastery of material.
  - Our industry partners want strong communication skills:

One More FAQ: Why Paper Planes?

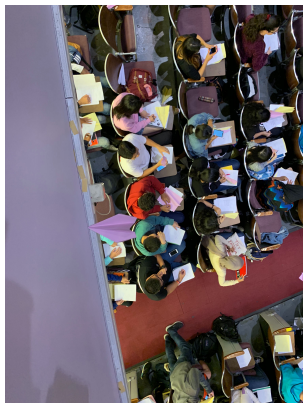


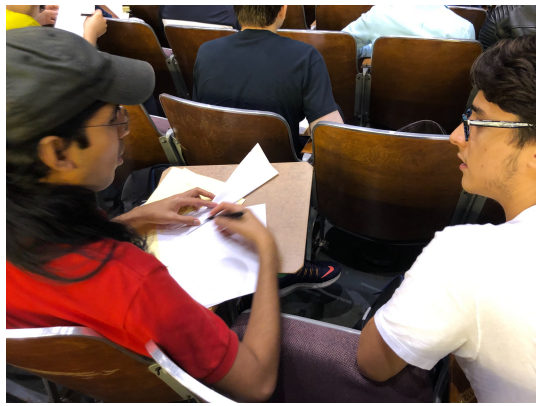
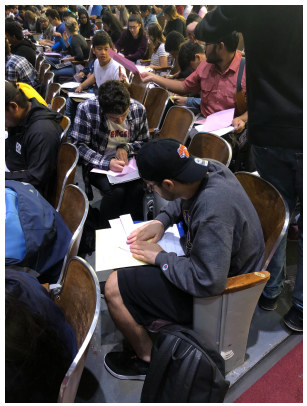



Why paper planes?

- It's a classic design question in introductory programming classes, since
  - Practice writing solutions (algorithms) in plain English without worrying about syntax (pseudocode).
  - Practice thinking (and writing) precisely.
- Why in groups?
  - Improves mastery of material.
  - Our industry partners want strong communication skills:
    - ★ communicating technical ideas precisely, and

One More FAQ: Why Paper Planes?



Why paper planes?

- It's a classic design question in introductory programming classes, since
  - Practice writing solutions (algorithms) in plain English without worrying about syntax (pseudocode).
  - Practice thinking (and writing) precisely.
- Why in groups?
  - Improves mastery of material.
  - Our industry partners want strong communication skills:
    - ★ communicating technical ideas precisely, and
    - ★ communicating and working in teams.

Plane Winners

Come claim your prizes after lecture:

Design Team:	*Build Team:*
Irene, Alisha, Charlie,	*(empty)*
(empty)	Shirley, Amanda
Kanglu, Ling, Xihao, Yaohoa	*(empty)*

Today's Topics

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

Today's Topics

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

In Pairs or Triples...

Some challenges with types & decisions:

```
#What are the types:
y1 = 2017
y2 = "2018"
print(type(y1))
print(type("y1"))
print(type(2017))
print(type("2017"))
print(type(y2))
print(type(y1/4.0))

x = int(y2) - y1
if x < 0:
    print(y2)
else:
    print(y1)
```

```
cents = 432
dollars = cents // 100
change = cents % 100
if dollars > 0:
    print('$'+str(dollars))
if change > 0:
    quarters = change // 25
    pennies = change % 25
    print(quarters, "quarters")
    print("and", pennies, "pennies")
```

Python Tutor

```
#What are the types:
y1 = 2017
y2 = "2018"
print(type(y1))
print(type("y1"))
print(type(2017))
print(type("2017"))
print(type(y2))
print(type(y1/4.0))

x = int(y2) - y1
if x < 0:
    print(y2)
else:
    print(y1)
```

(Demo with pythonTutor)

Decisions

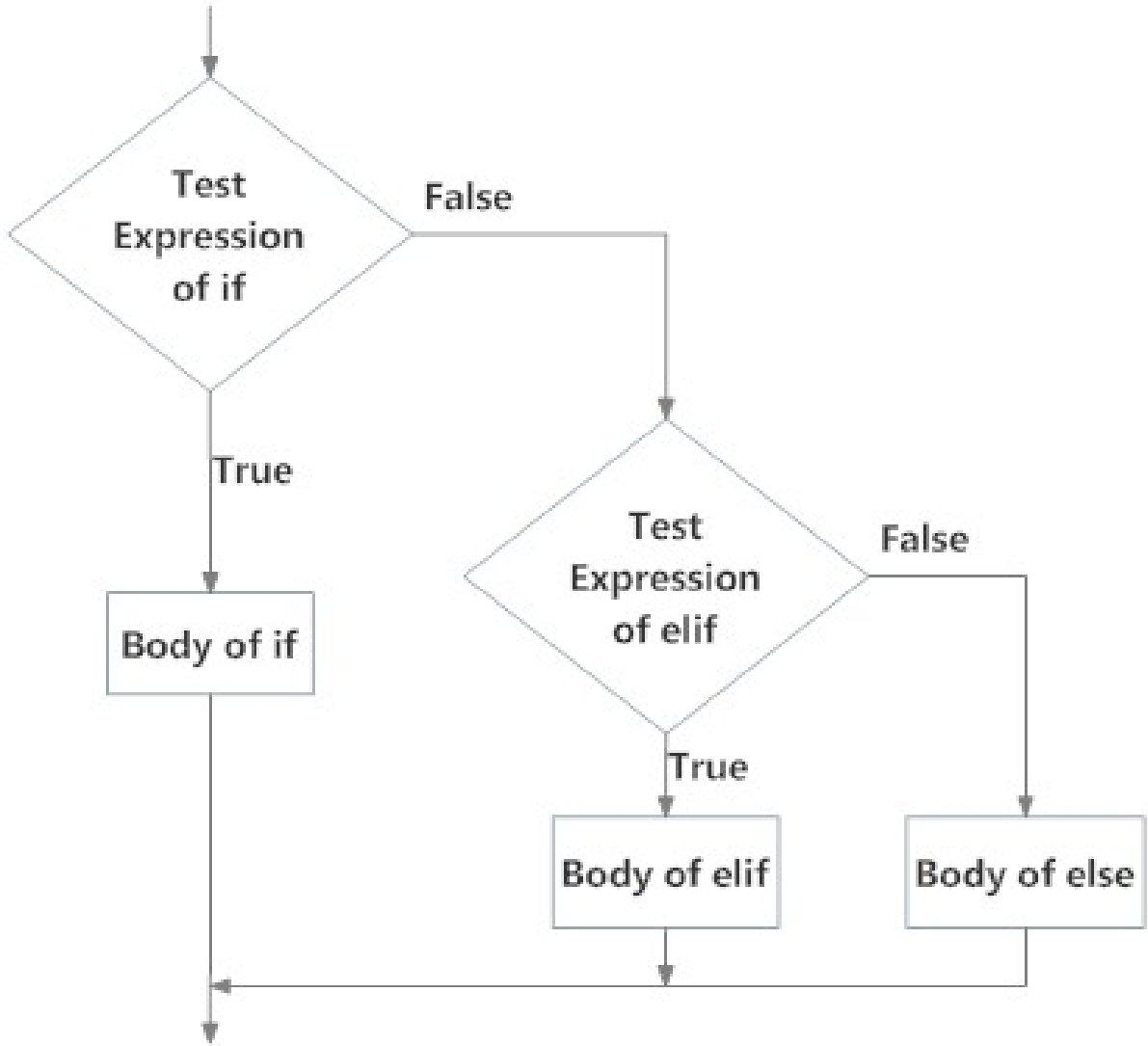
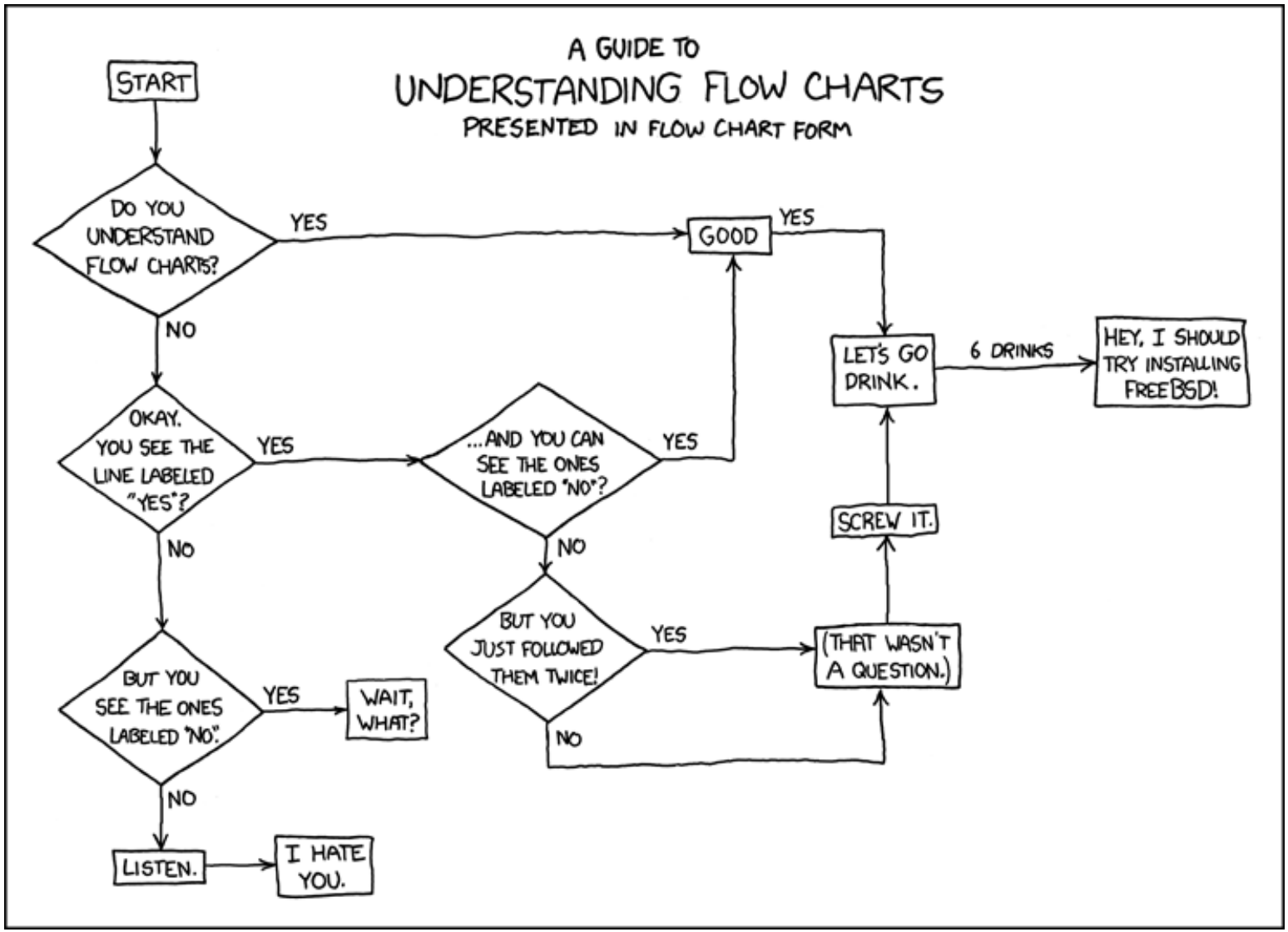


Fig: Operation of if...elif...else statement

(programiz)

Side Note: Reading Flow Charts



(xkcd/518)

Today's Topics

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

In Pairs or Triples

Predict what the code will do:

```
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
      (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

Python Tutor

```
origin = "Indian Ocean"
winds = 100
if (winds > 74):
    print("Major storm, called a ", end="")
    if origin == "Indian Ocean" or origin == "South Pacific":
        print("cyclone.")
    elif origin == "North Pacific":
        print("typhoon.")
    else:
        print("hurricane.")

visibility = 0.2
winds = 40
conditions = "blowing snow"
if (winds > 35) and (visibility < 0.25) and \
     (conditions == "blowing snow" or conditions == "heavy snow"):
    print("Blizzard!")
```

(Demo with `pythonTutor`)

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`

In Pairs or Triples

Predict what the code will do:

```
semHours = 18
reqHours = 120
if semHours >= 12:
    print('Full Time')
else:
    print('Part Time')

pace = reqHours // semHours
if reqHours % semHours != 0:
    pace = pace + 1
print('At this pace, you will graduate in', pace, 'semesters,')
yrs = pace / 2
print('(or', yrs, 'years).')


for i in range(1,20):
    if (i > 10) and (i % 2 == 1):
        print('oddly large')
    else:
        print(i)
```

Python Tutor

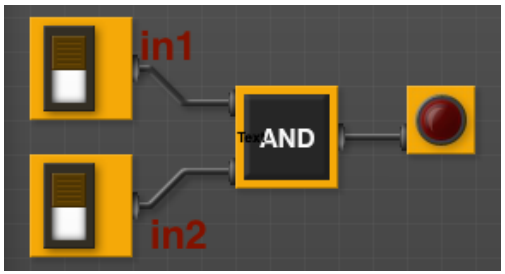
```
semHours = 18
reqHours = 120
if semHours >= 12:
     print('Full Time')
else:
     print('Part Time')

pace = reqHours // semHours
if reqHours % semHours != 0:
     pace = pace + 1
print('At this pace, you will graduate in', pace, 'semesters,')
yrs = pace / 2
print('(or', yrs, 'years).')


for i in range(1,20):
     if (i > 10) and (i % 2 == 1):
          print('oddly large')
     else:
          print(i)
```

(Demo with pythonTutor)

Today's Topics

- Recap: Decisions
- Logical Expressions
- **Circuits**
- CS Survey

Circuit Demo

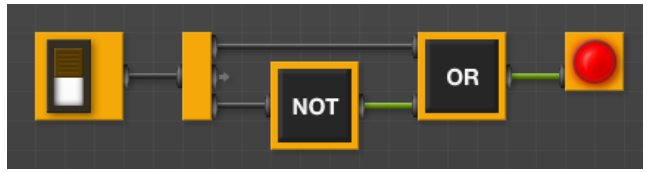


(Demo with `neuroproductions`)

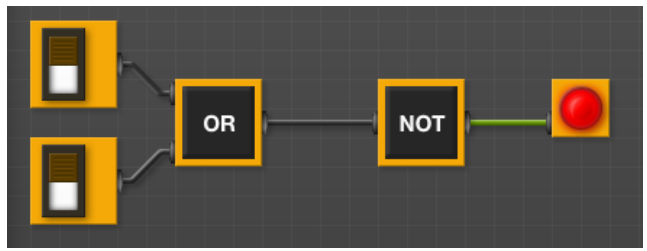
In Pairs or Triples

Predict when these expressions are true:

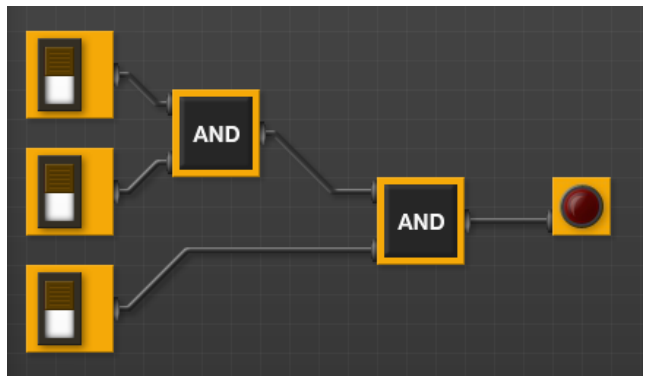
- `in1 or not in1`:



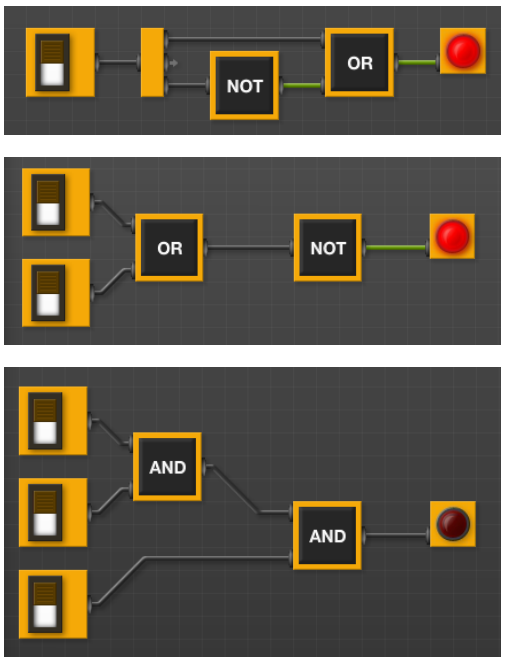
- `not(in1 or in2)`:



- `(in1 and in2) and in3`:

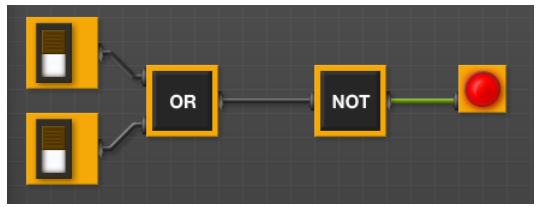


Circuit Demo



(Demo with `neuroproductions`)

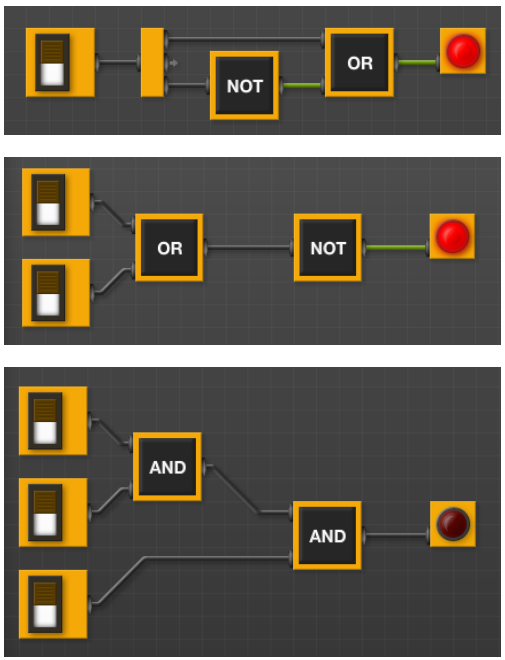
In Pairs or Triples



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 `neuroproductions`)

Today's Topics

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

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

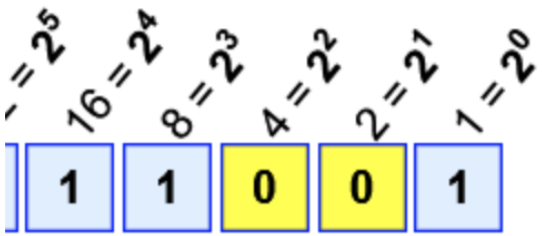
Binary Numbers

- 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)

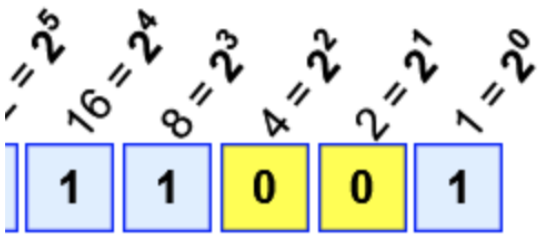
Binary Numbers



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

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

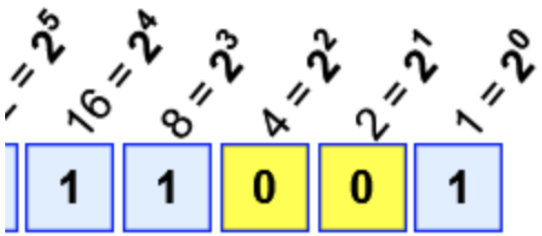
Binary Numbers



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

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

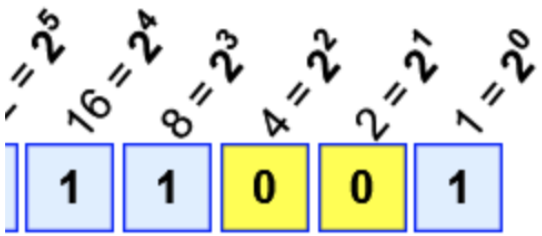
Binary Numbers



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)

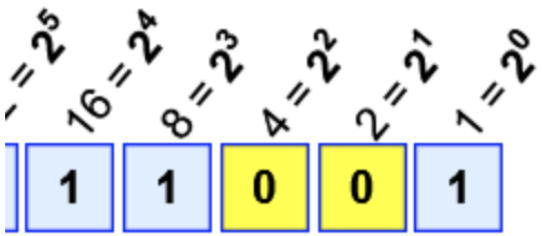
Binary Numbers



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)

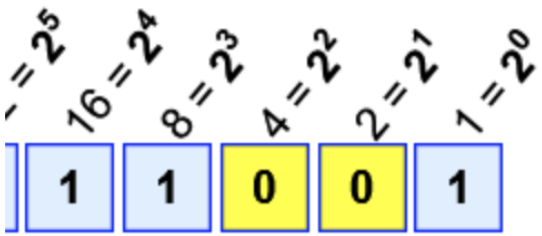
Binary Numbers



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 we can have either a 0 or a 1

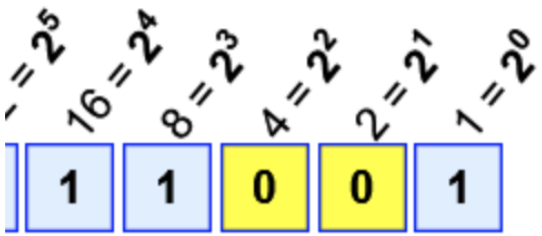
Binary Numbers



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

- Two digits: **0** and **1**
- Each position is a power of two
  - 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 we can have either a 0 or a 1
  - In the "ones" position we either have a 1 or not

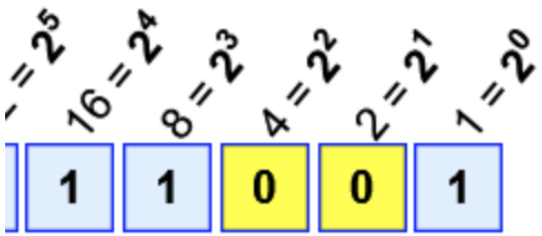
Binary Numbers



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 we can have either a 0 or a 1
  - In the "ones" position we either have a 1 or not
  - In the "twos" position we either have a 2 or not

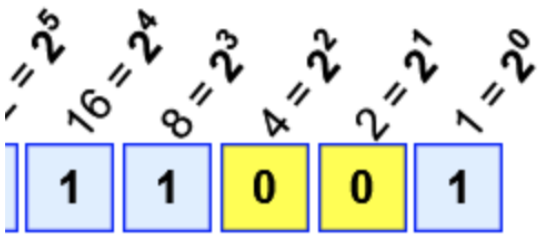
Binary Numbers



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 we can have either a 0 or a 1
  - 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 we can have either a 0 or a 1
  - 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:**

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

Today's Topics

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

CS Survey Talk: CUNY2X & TTP @Hunter

Bernard Desert & Elise Harris

CS Survey Talk: CUNY2X & TTP @Hunter

Bernard Desert & Elise Harris

- Brief overview of CUNY 2X & Tech Talent Pipeline

CS Survey Talk: CUNY2X & TTP @Hunter

Bernard Desert & Elise Harris

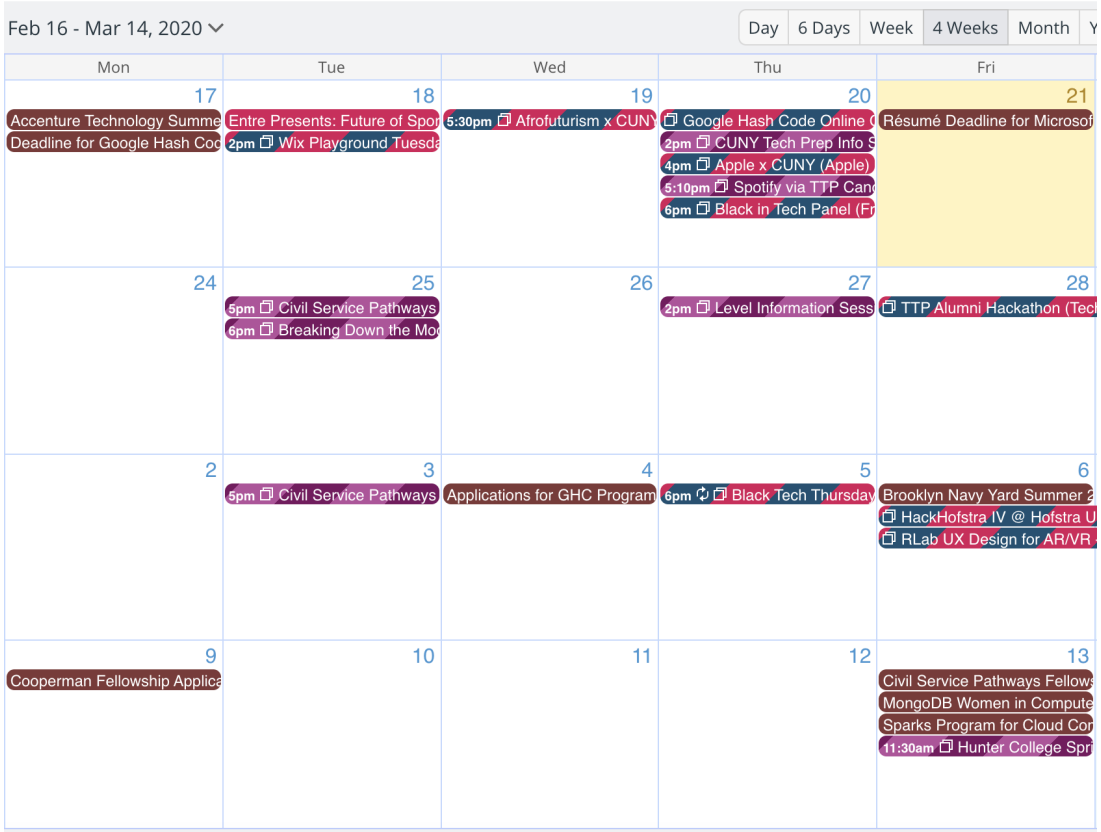
- Brief overview of CUNY 2X & Tech Talent Pipeline
- What Bernard & Elise love about their jobs.

CS Survey Talk: CUNY2X & TTP @Hunter

Bernard Desert & Elise Harris

- Brief overview of CUNY 2X & Tech Talent Pipeline
- What Bernard & Elise love about their jobs.
- Design challenge: classic tech interview question.

CS Survey Talk: Hunter Tech Calendar



Sign up:

- Tech events calendar: `http://bit.ly/HunterTechCalendar`
- Newsletter: `http://bit.ly/CUNY2XNewsletter`
- Hunter CS Handbook: `http://bit.ly/huntercshandbook`

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1
  2

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1
  2
  Fizz

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1
  2
  Fizz
  4

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1
  2
  Fizz
  4
  Buzz

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1
  2
  Fizz
  4
  Buzz
  5

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1
  2
  Fizz
  4
  Buzz
  5
  Fizz

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- Write down the output to see the pattern:
  1
  2
  Fizz
  4
  Buzz
  5
  Fizz
  7

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".
- Write down the output to see the pattern:
  1
  2
  Fizz
  4
  Buzz
  5
  Fizz
  7
  ...
  14

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".
- Write down the output to see the pattern:
  1
  2
  Fizz
  4
  Buzz
  5
  Fizz
  7
  ...
  14
  FizzBuzz

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List:

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List:
  - Create a loop that goes from 1 to 100.

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List:
  - Create a loop that goes from 1 to 100.
  - If the number is divisible by 3, print “Fizz”.

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List:
  - Create a loop that goes from 1 to 100.
  - If the number is divisible by 3, print “Fizz”.
  - If the number is divisible by 5, print “Buzz”.

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List:
  - Create a loop that goes from 1 to 100.
  - If the number is divisible by 3, print “Fizz”.
  - If the number is divisible by 5, print “Buzz”.
  - If divisible by both, print “FizzBuzz”.

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List:
  - Create a loop that goes from 1 to 100.
  - If the number is divisible by 3, print “Fizz”.
  - If the number is divisible by 5, print “Buzz”.
  - If divisible by both, print “FizzBuzz”.
  - **Otherwise print the number.**

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List:
  - Create a loop that goes from 1 to 100.
  - If the number is divisible by 3, print “Fizz”.
  - If the number is divisible by 5, print “Buzz”.
  - If divisible by both, print “FizzBuzz”.
  - **Otherwise print the number.**
    We should do this one first!

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.
- To Do List (**Reordered**):

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".
- To Do List (**Reordered**):
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".

Tech Interview Classic

- Write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".
- To Do List (**Reordered**):
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print “Fizz”.
  - If the number is divisible by 5, print “Buzz”.
  - If divisible by both, print “FizzBuzz”.
  - Also should print a new line (so each entry is on its own line).

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
    if i%3 != 0 and i%5 != 0:
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
    if i%3 != 0 and i%5 != 0:
        print(i, end="")
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
    if i%3 != 0 and i%5 != 0:
        print(i, end="")
    if i%3 == 0:
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
    if i%3 != 0 and i%5 != 0:
        print(i, end="")
    if i%3 == 0:
        print("Fizz", end="")
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
    if i%3 != 0 and i%5 != 0:
        print(i, end="")
    if i%3 == 0:
        print("Fizz", end="")
    if i%5 == 0:
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
    if i%3 != 0 and i%5 != 0:
        print(i, end="")
    if i%3 == 0:
        print("Fizz", end="")
    if i%5 == 0:
        print("Buzz", end="")
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - Print the numbers not divisible by 3 or 5.
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - If divisible by both, print "FizzBuzz".
  - Also should print a new line (so each entry is on its own line).
- One solution (uses `print(,end="")` that prints all on the same line):

```
for i in range(1,101):
    if i%3 != 0 and i%5 != 0:
        print(i, end="")
    if i%3 == 0:
        print("Fizz", end="")
    if i%5 == 0:
        print("Buzz", end="")
    print()
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

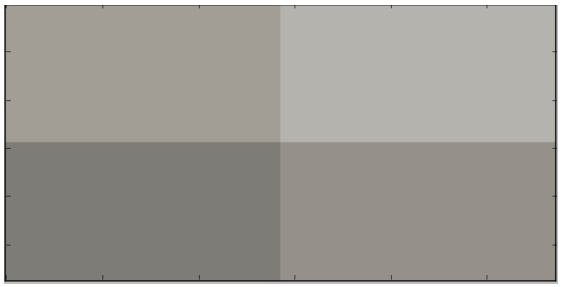
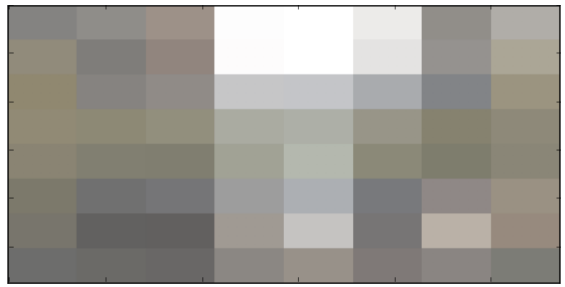
Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
  - Decisions
  - Logical Expressions
  - Circuits
  - Binary Numbers

Recap

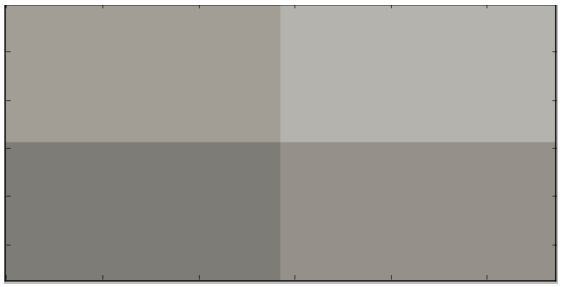
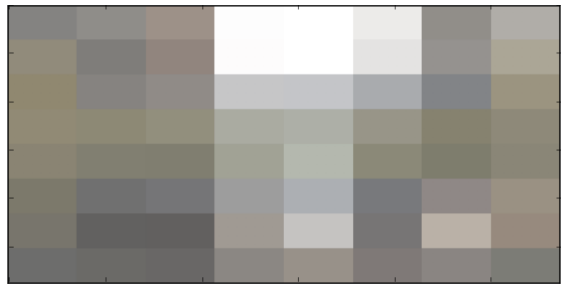
- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
  - Decisions
  - Logical Expressions
  - Circuits
  - Binary Numbers
- Pass your lecture slips to the aisles for the UTAs to collect.

Practice Quiz & Final Questions

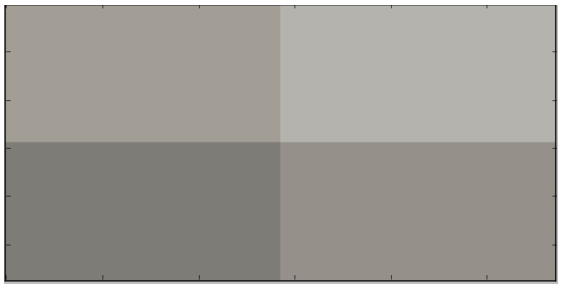
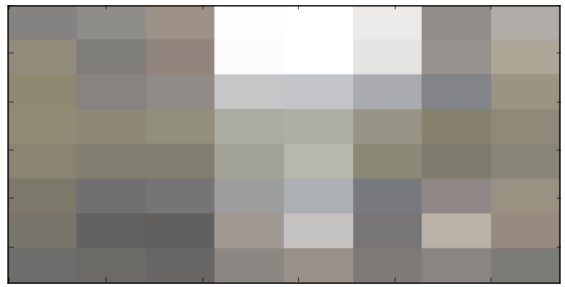

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.

Practice Quiz & Final Questions

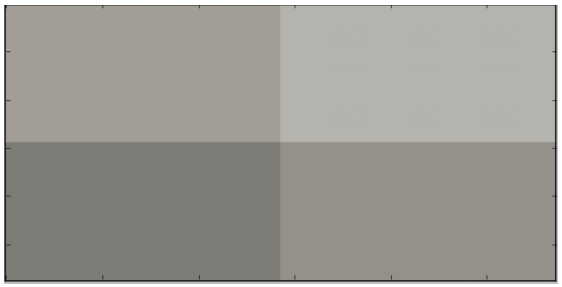
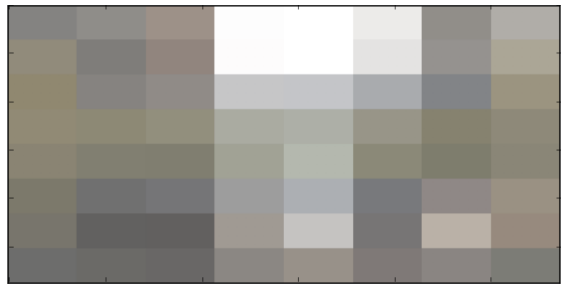

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).

Practice Quiz & Final Questions

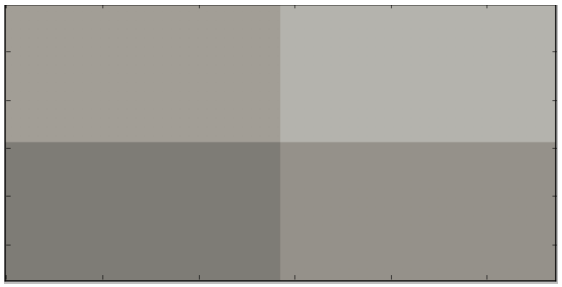
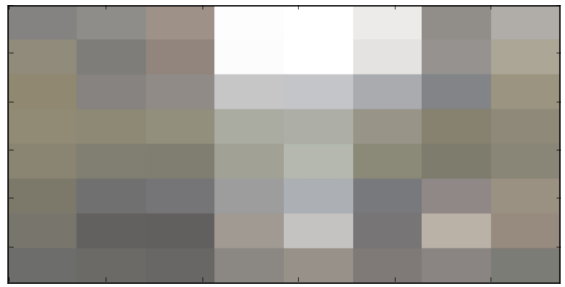

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:

Practice Quiz & Final Questions

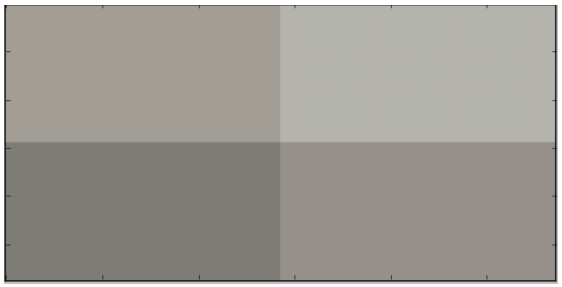
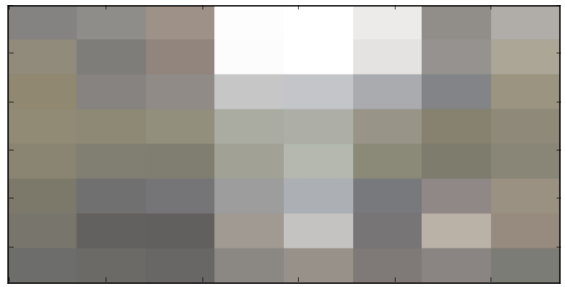

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;

Practice Quiz & Final Questions

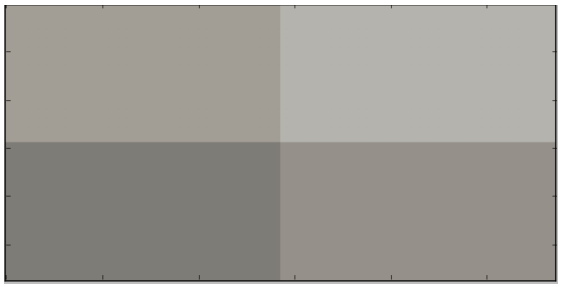
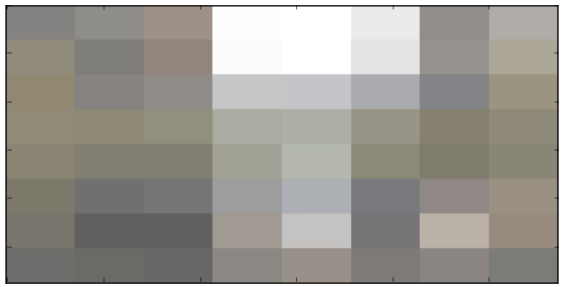
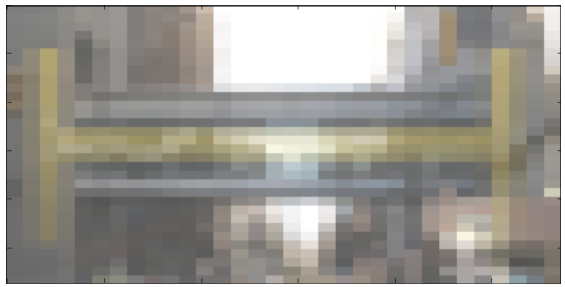

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and

Practice Quiz & Final Questions

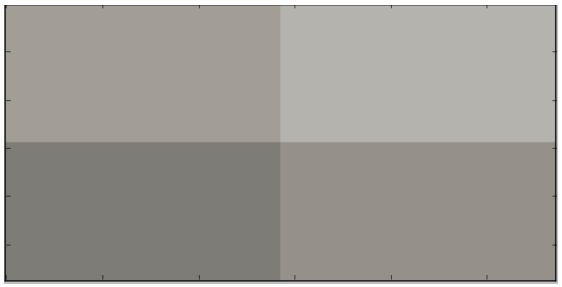
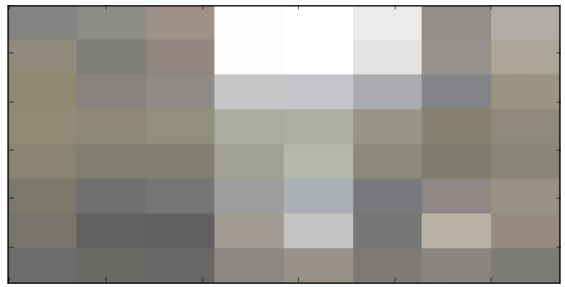

- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.

Practice Quiz & Final Questions

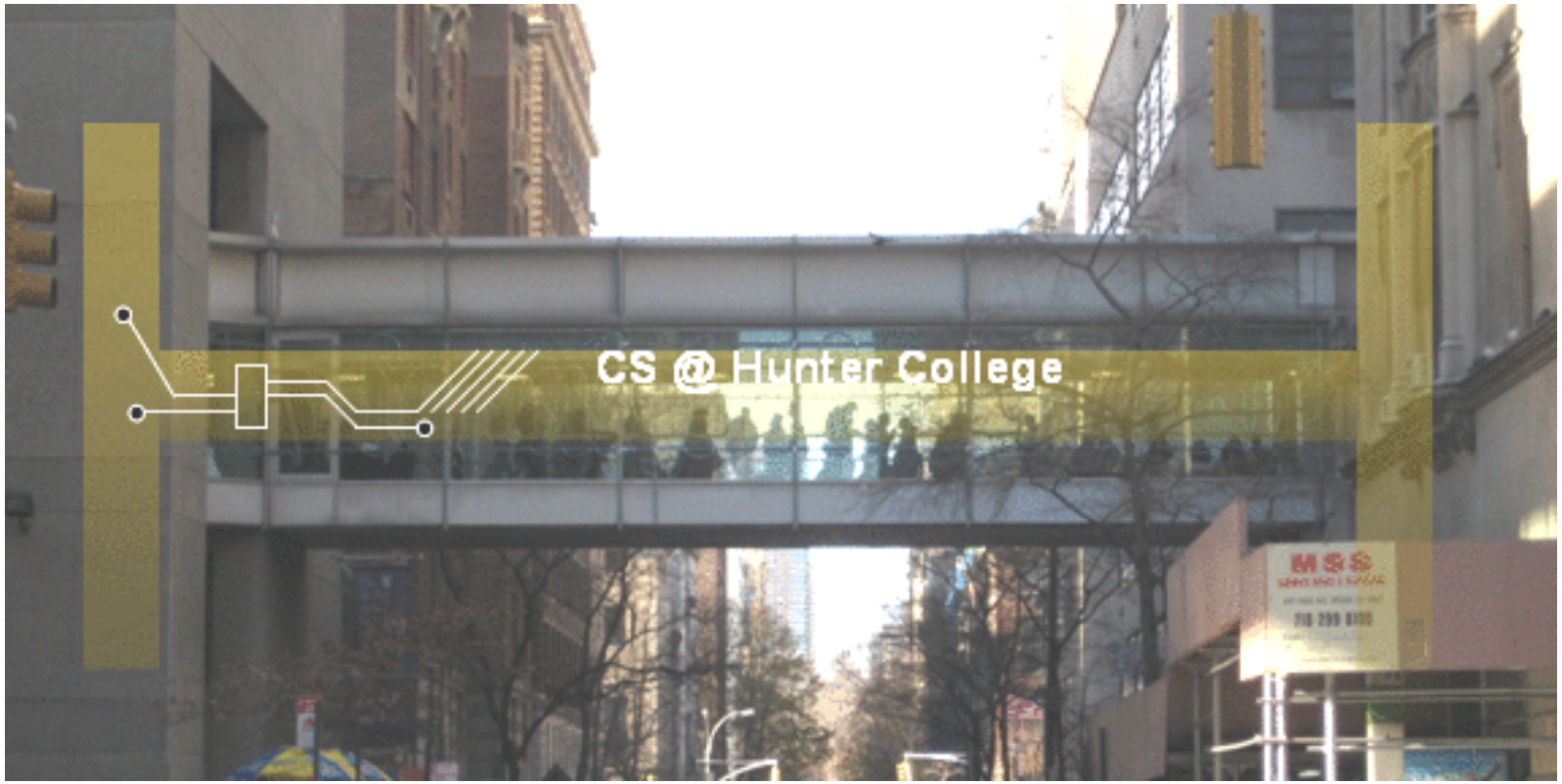
- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final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

Practice Quiz & Final Questions


- 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).
- We're starting with Spring 2018, Version 1.

Writing Boards



- Return writing boards as you leave...