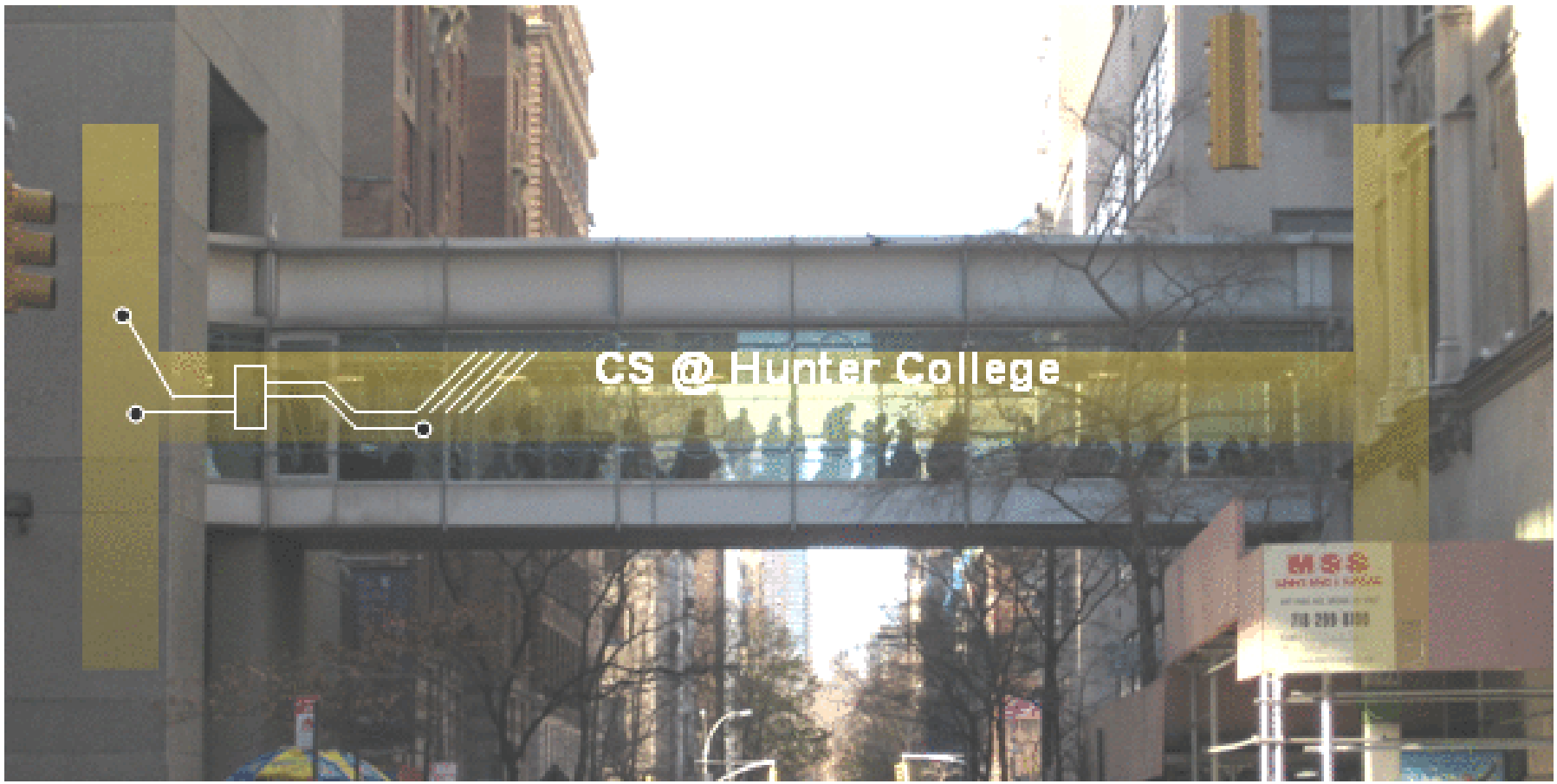
CSci 127: Introduction to Computer Science



Finished the lecture preview?

hunter.cuny.edu/csci

Announcement: Academic Dishonesty

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

Announcement: Academic Dishonesty

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

- **All instances of academic dishonesty will be reported to the office of student affairs.**

Announcement: Academic Dishonesty

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

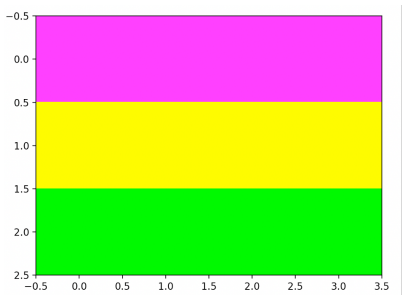
- **All instances of academic dishonesty will be reported to the office of student affairs.**

Represent numbers by different colors to get an image

For example, given a text file with contents

```
1 1 1 1
2 2 2 2
3 3 3 3
```

We would like to get an image as follows.



To test online,

1. Go to trinket.
2. Upload text file using upload button in top right.
3. Edit code in main.py. Click Run button.

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 #link to program
4 inF = input("Enter a file name: ")
5 data = np.loadtxt(inF)
```

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 #link to program
4 inF = input("Enter a file name: ")
5 data = np.loadtxt(inF)
6
7 dimensions = data.shape + (3,) #add a layer of size 3 to represent r, g, b
      channels
```

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 #link to program
4 inF = input("Enter a file name: ")
5 data = np.loadtxt(inF)
6
7 dimensions = data.shape + (3,) #add a layer of size 3 to represent r, g, b
      channels
8 img = np.zeros(dimensions)
```

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  #link to program
4  inF = input("Enter a file name: ")
5  data = np.loadtxt(inF)
6
7  dimensions = data.shape + (3,) #add a layer of size 3 to represent r, g, b
       channels
8  img = np.zeros(dimensions)
9
10 for row in range(data.shape[0]):
11     for col in range(data.shape[1]):
```

```
import matplotlib.pyplot as plt
import numpy as np
#link to program
inF = input("Enter a file name: ")
data = np.loadtxt(inF)

dimensions = data.shape + (3,) #add a layer of size 3 to represent r, g, b
    channels
img = np.zeros(dimensions)

for row in range(data.shape[0]):
    for col in range(data.shape[1]):
        if data[row, col] == 1:
            img[row, col, ... ] = ... #purple
```

```
import matplotlib.pyplot as plt
import numpy as np
#link to program
inF = input("Enter a file name: ")
data = np.loadtxt(inF)

dimensions = data.shape + (3,) #add a layer of size 3 to represent r, g, b
    channels
img = np.zeros(dimensions)

for row in range(data.shape[0]):
    for col in range(data.shape[1]):
        if data[row, col] == 1:
            img[row, col, ... ] = ... #purple
        elif data[row, col] == 2:
            img[row, col, ...] = ... #yellow
```

```
import matplotlib.pyplot as plt
import numpy as np
#link to program
inF = input("Enter a file name: ")
data = np.loadtxt(inF)

dimensions = data.shape + (3,) #add a layer of size 3 to represent r, g, b
    channels
img = np.zeros(dimensions)

for row in range(data.shape[0]):
    for col in range(data.shape[1]):
        if data[row, col] == 1:
            img[row, col, ... ] = ... #purple
        elif data[row, col] == 2:
            img[row, col, ...] = ... #yellow
        elif data[row, col] == 3:
            img[row, col, ...] = ... #green

plt.imshow(img)
plt.show()
```

Today's Topics

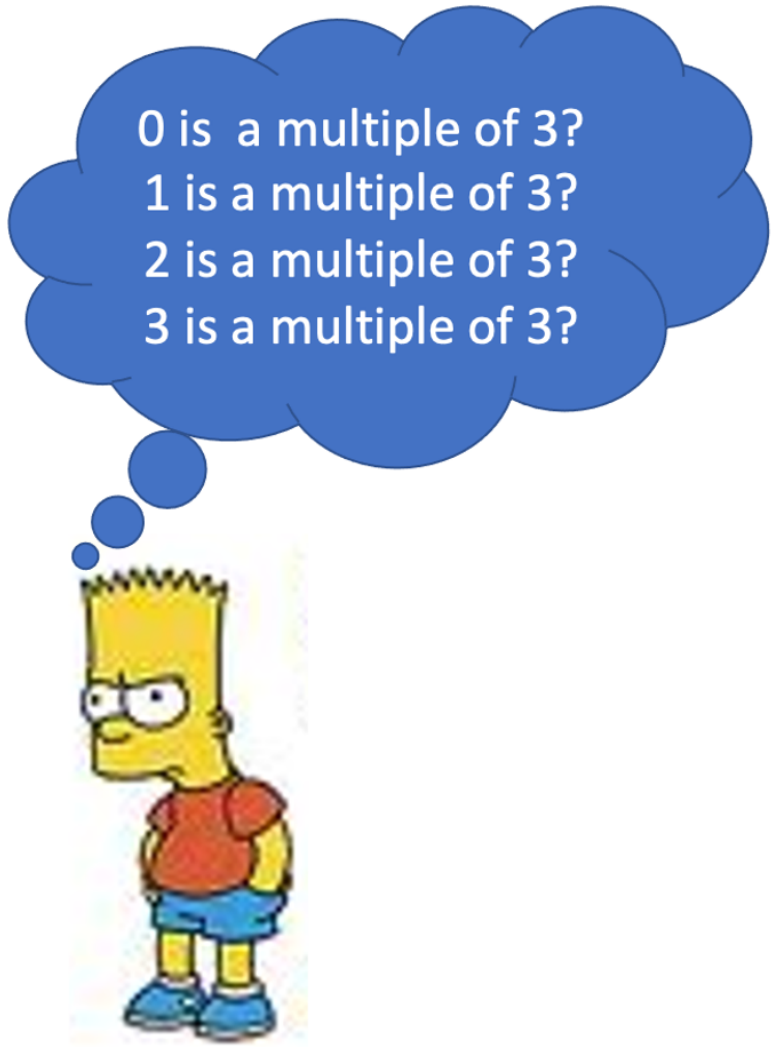
- **Recap: Decisions**
- Logical Expressions
- Circuits
- Binary Numbers
- Design Challenge: Airplanes

A story about if statement

Ann: **If** you have \$1000, will you please give me a half?
Bob: Of course.
Ann: **If** you have \$100, will you please give me a half?
Bob: Sure.
Ana: **If** you have \$10, will you please give me a half?
Bob: **NO WAY!!**
Ana: Why?
Bob: I do NOT have \$100 or more, but I do have \$10.

An example of if statement

Enter an int, find out whether it is a multiple of 3?



Code to find out whether an input is a multiple of 3

Input an int, if it is a multiple of 3, print that this number is a multiple of 3, otherwise, do nothing.
What is the output when input is 0?
What is the output when input is 2?
What is the output when input is 3?

```
1 numStr = input("Enter an int: ")
2 num = int(numStr)
3 #can replace the above two statements as
4 #num = int(input("Enter an int: "))
5
6 if num % 3 == 0:
7     print(num, "is a multiple of 3")
```

Code to find out whether an input is a multiple of 3 or not

Input an int, if it is a multiple of 3, print that this number is a multiple of 3, otherwise, print it is not a multiple of 3.
What is the output when input is 0?
What is the output when input is 2?
What is the output when input is 3?

```
1 numStr = input("Enter an int: ")
2 num = int(numStr)
3 #can replace the above two statements as
4 #num = int(input("Enter an int: "))
5
6 if num % 3 == 0:
7     print(num, "is a multiple of 3")
8 else:
9     print(num, "is not a multiple of 3")
```

Nested if-else statements: handle more than two cases

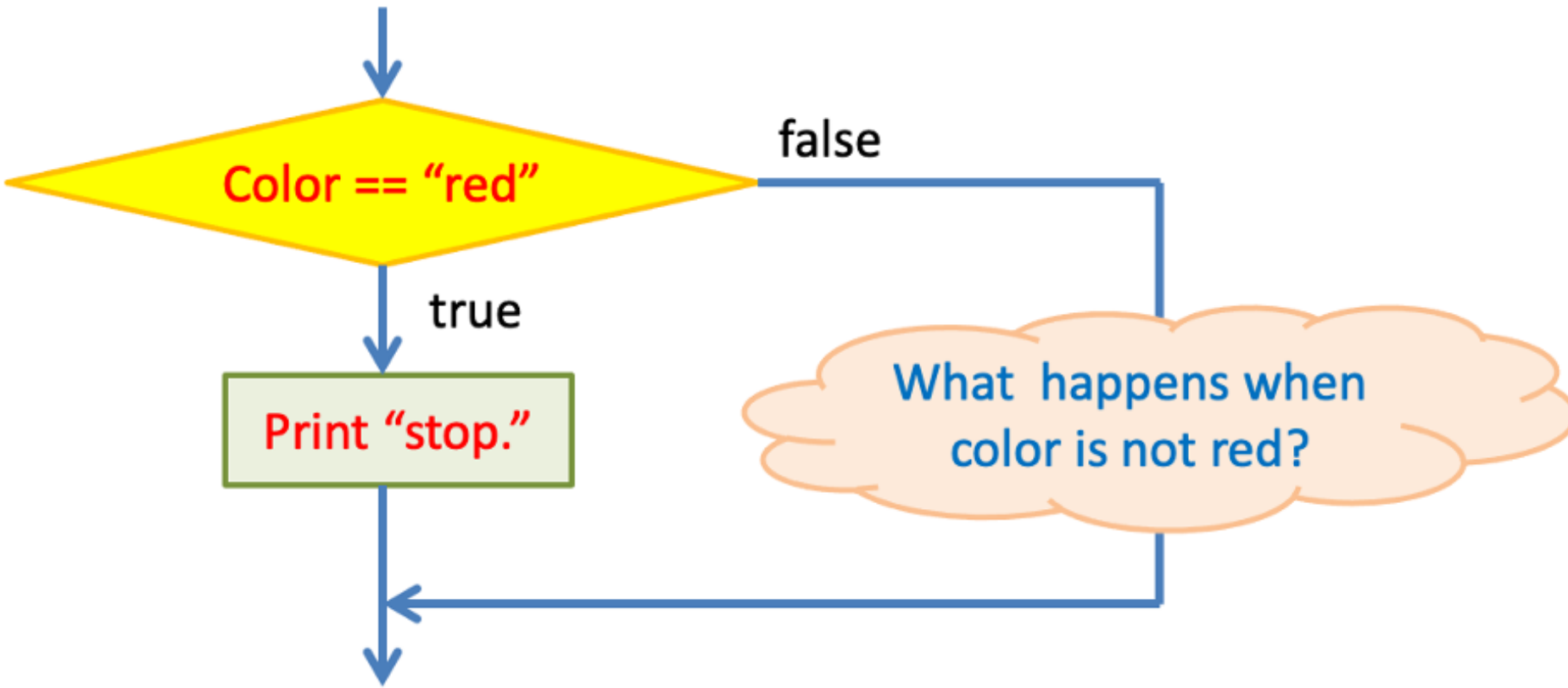
- In an exam, we may have only two outcomes (pass or fail).
- Sometimes, life has more than two possibilities. For example,
  - Signals of a traffic light
  - Even an exam can have A, B, C, D, F grades.
  - Taxes for different household incomes

Traffic Light

Enter a string representing color (red, green, yellow),

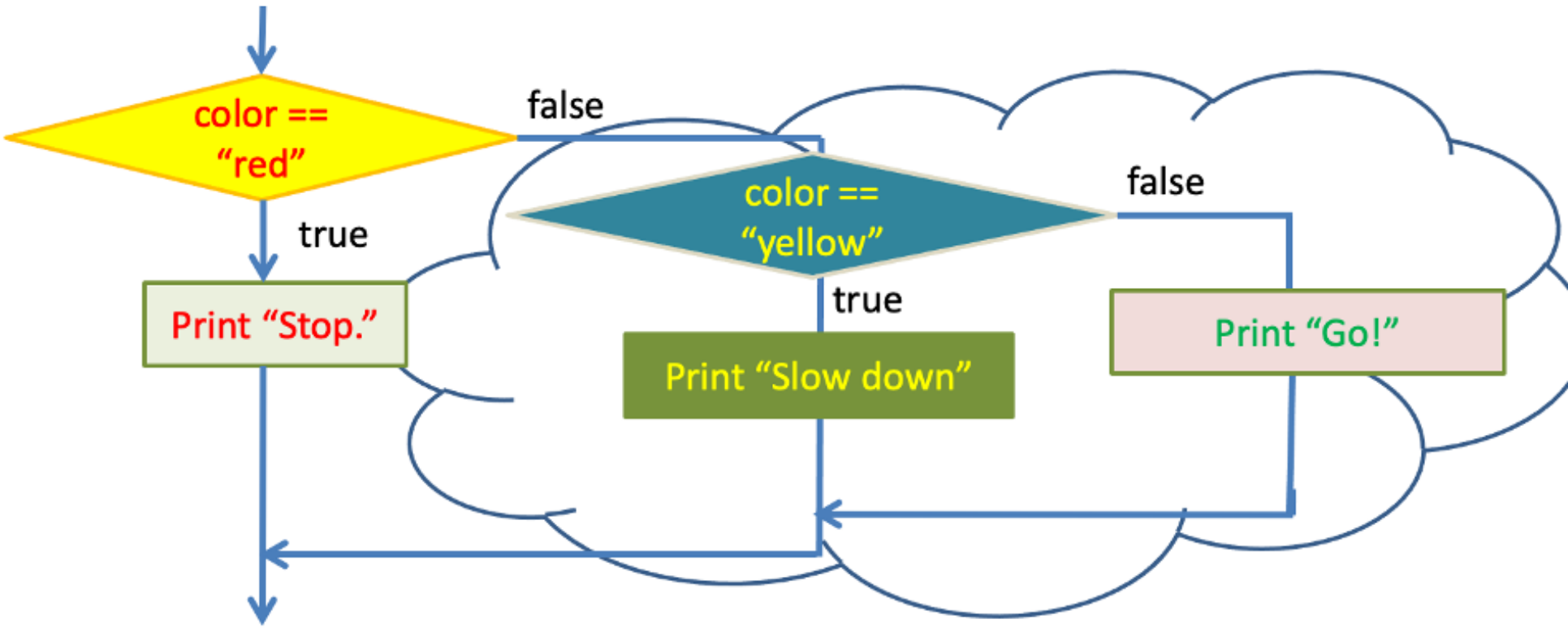
- print "Stop" if the color is red,
- print "Go" if the color is green,
- print "Slow down" if the color is yellow.

What if color is red? Use == to compare two items equal or not.



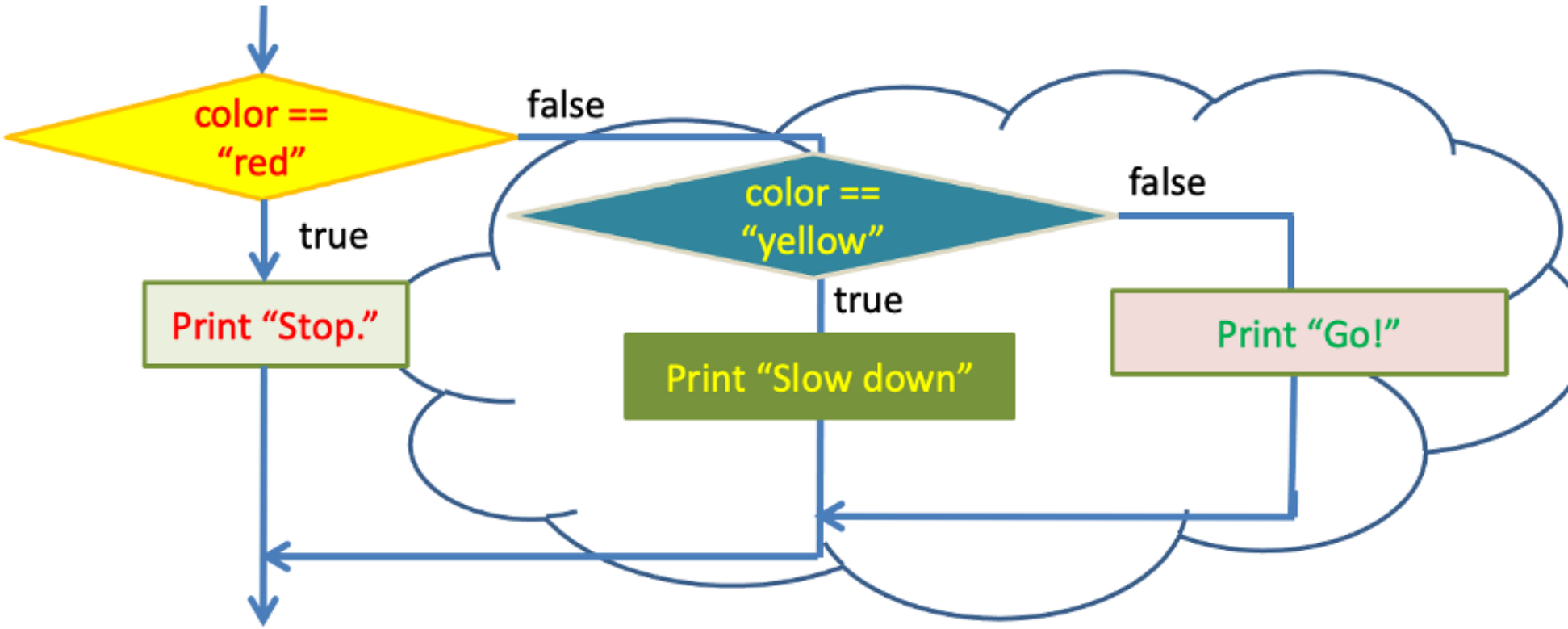
Traffic Light: II

What if color is not red but yellow?



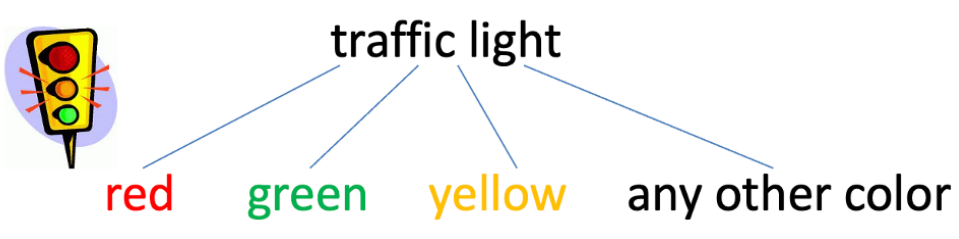
Traffic Light: III

Suppose the available colors are red, green, and yellow only. What if the color is neither red nor yellow?



Traffic Light: III

What if the color is not one of the following: red, yellow, green?




```
1 color = input("Enter traffic light color (red, green,
      yellow): ") #link
2 if color == "red":
3     print("stop")
4 elif color == "green":
5     print("go")
6 elif color == "yellow":
7     print("slow down")
8 else:
9     print("wrong color")
```

Common mistakes in writing a nested if-else statement

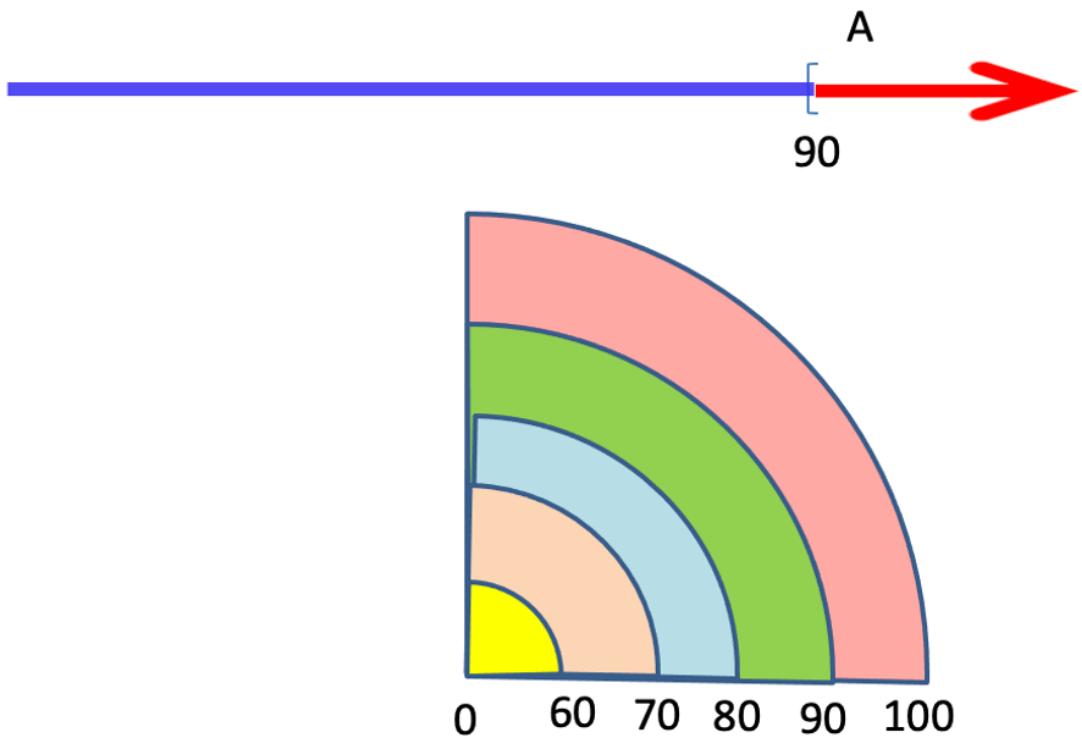
Fix all errors.

```
1 color = input("Enter traffic light color (red, green,
      yellow): ")
2 if color = 'red": #== equal comparison, = assignment
3     print("stop")
4 if color == "green"
5     print("go")
6 if color == "yellow":
7 print("slow down")
8 else:
9     print("wrong color")
```

Convert numerical grade to letter grade

Enter numerical grade, if it is larger than equal to 90, print "A", else if it is larger than or equal to 80, print "B", else if it is larger than or equal to 70, print "C", else if it is larger than or equal to 60, print "D", else print "F".

Peel an onion



Challenge 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")
```

Demo with Python Tutor

Challenge - *Predict what the code will do:*

```
semHours = 18 #link to python Tutor
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)
```

Today's Topics

- Recap: Decisions
- **Logical Expressions**
- Circuits
- Binary Numbers
- Design Challenge: Airplanes

Logical Operators

1. in1 and in2 is True when both in1 and in2 are True.
2. in1 or in2 is True when at least one of in1 or in2 is True.
3. Not True is False, not False is True.

and

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

Logical Operators

1. in1 and in2 is True when both in1 and in2 are True.
2. in1 or in2 is True when at least one of in1 or in2 is True.
3. Not True is False, not False is True.

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

1. in1 and in2 is True when both in1 and in2 are True.
2. in1 or in2 is True when at least one of in1 or in2 is True.
3. Not True is False, not False is True.

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

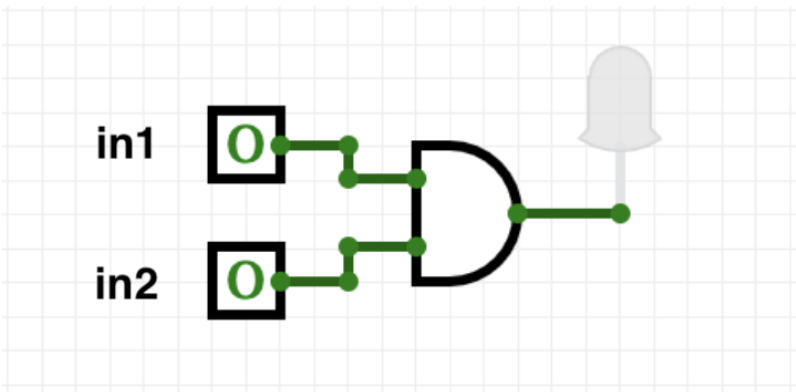
not

	in1	*returns:*
not	False	True
not	True	False

Today's Topics

- Recap: Decisions
- Logical Expressions
- **Circuits**
- Binary Numbers
- Design Challenge: Airplanes

Circuit Demo

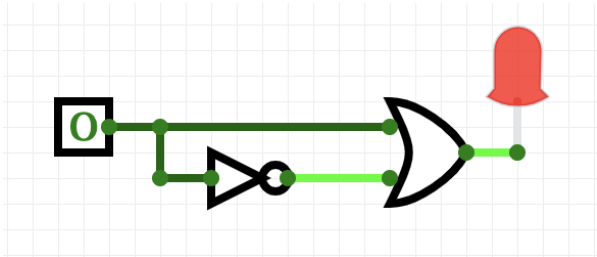


(Demo with `and-gate circuitverse`

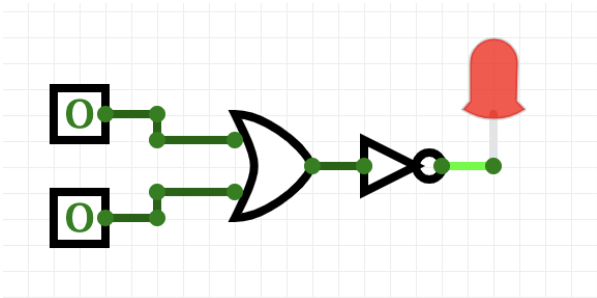
Challenge

Predict when these expressions are true:

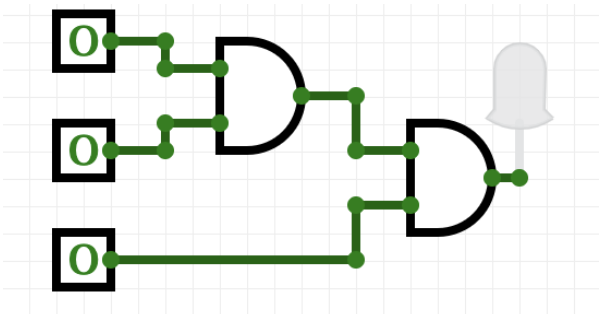
- `in1 or not in1`:



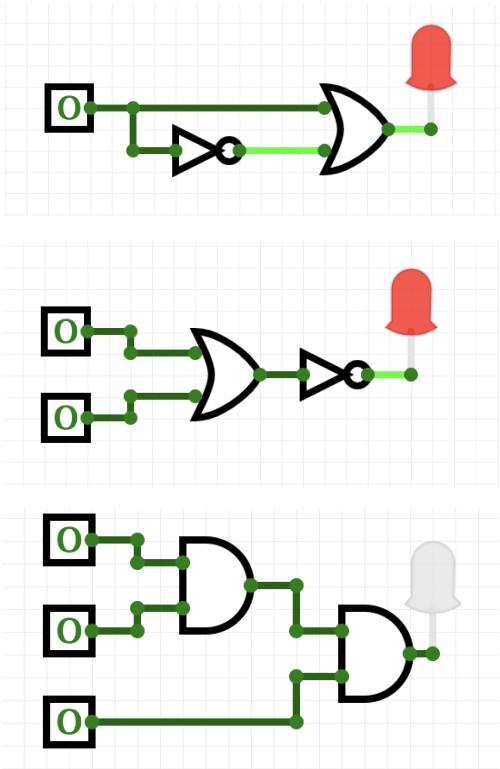
- `not(in1 or in2)`:



- `(in1 and in2) and in3`:

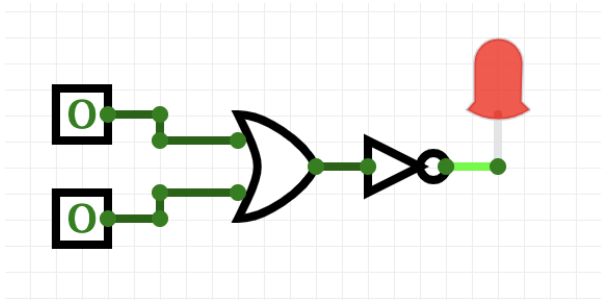


Circuit Demo



(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))`

Today's Topics

- Recap: Decisions
- Logical Expressions
- Circuits
- **Binary Numbers**
- Design Challenge: Airplanes

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

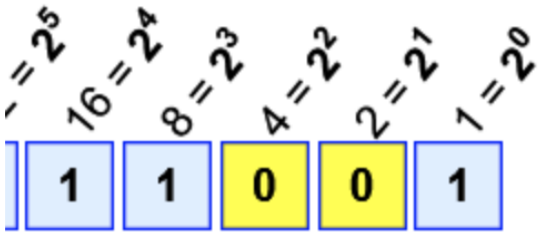
Binary Numbers

- 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)

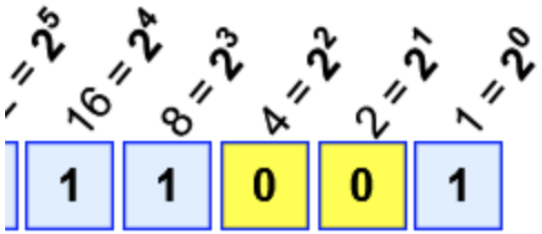
Binary Numbers



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

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

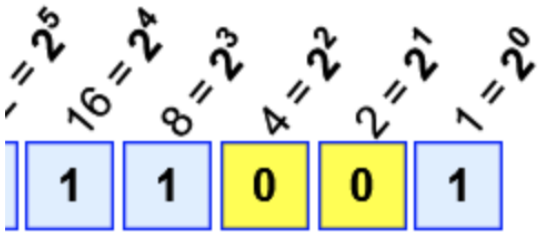
Binary Numbers



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

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

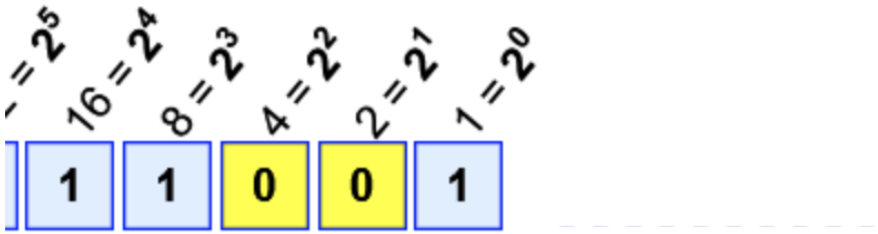
Binary Numbers



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)

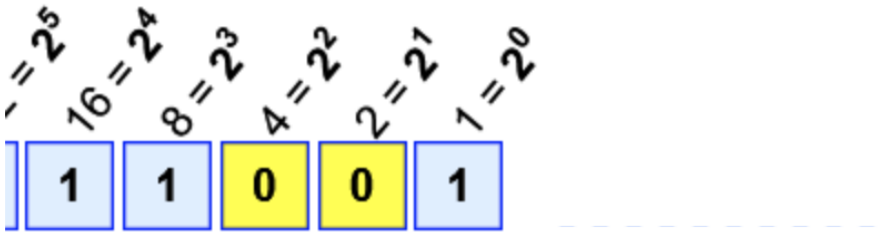
Binary Numbers



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)

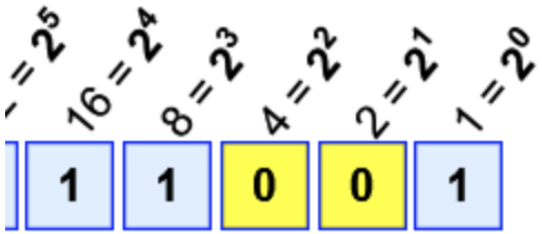
Binary Numbers



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:

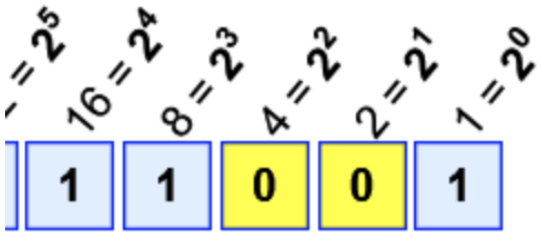
Binary Numbers



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

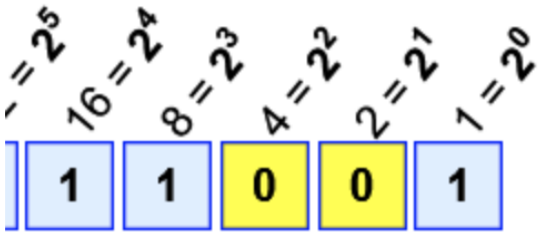
Binary Numbers



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

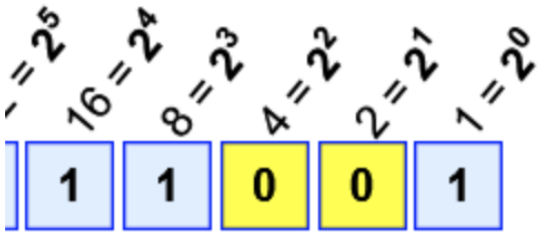
Binary Numbers



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:**

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

Decimal and Binary

Related: video for hexadecimal, start from time 17:53

	decimal	binary
base	10	2
digits	0-9	0-1
eg	$26 = 2 * 10^1 + 6 * 10^0$	$11010 = 1 * 2^4 + 1 * 2^3 + 1 * 2^1 = 16 + 8 + 2 = 26_{10}$
	$255 = 2 * 10^2 + 5 * 10^1 + 5 * 10^0$	$11111111_2 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255_{10}$

Steps to convert binary to decimal

1. Start from rightmost to leftmost digit, label exponent as 0, 1, 2, ...
2. When digit is 1, multiple each digit by $base^{exponent}$, where base is 2 for binary numbers.
3. Add the products in the second step up.

Convert Decimal to Binary

1. Divide the number by base 2. Calculate quotient and remainder.
2. Set the quotient to be number. Repeat the above step until quotient is zero.
3. Connect the remainders **backwards**.

```
convert to decimal
10 |  26
   +-----
10 |   2    6   ^
   +-----       |
       0    2   |
```

```
convert to binary
2 |  26
  +-----
2 |  13      0   ^
  +-----        |
2 |   6      1   |
  +-----        |
2 |   3      0   |
  +-----        |
2 |   1      1   |
  +-----        |
      0      1   |
```

Lecture Slip Challenge: 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”.

Lecture Slip Challenge: 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:

Lecture Slip Challenge: 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

Lecture Slip Challenge: 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

Lecture Slip Challenge: 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

Lecture Slip Challenge: 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

Lecture Slip Challenge: 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

Lecture Slip Challenge: 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
  Fizz

Lecture Slip Challenge: 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
  Fizz
  7

Lecture Slip Challenge: 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
  Fizz
  7
  ...
  14

Lecture Slip Challenge: 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
  Fizz
  7
  ...
  14
  FizzBuzz

Lecture Slip Challenge: 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
  Fizz
  7
  ...
  14
  FizzBuzz
- Write the **algorithm** then, if time, write the code.

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.
    Order matters!!! To print FizzBuzz when i is divisible by both it should be checked first, otherwise it will never get to this case!

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.
  - **If divisible by both 3 and 5, 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.
  - **If divisible by both 3 and 5, print "FizzBuzz".**
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - Otherwise print the number.
  - 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.
  - If divisible by both 3 and 5, print “FizzBuzz”.
  - If the number is divisible by 3, print “Fizz”.
  - If the number is divisible by 5, print “Buzz”.
  - Otherwise print the number.
  - 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.
  - If divisible by both 3 and 5, print "FizzBuzz".
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - Otherwise print the number.
  - Also should print a new line (so each entry is on its own line).

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

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - If divisible by both 3 and 5, print "FizzBuzz".
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - Otherwise print the number.
  - Also should print a new line (so each entry is on its own line).

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

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - If divisible by both 3 and 5, print "FizzBuzz".
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - Otherwise print the number.
  - Also should print a new line (so each entry is on its own line).

```
1 for i in range(1,101):
2     if i%3 == 0 and i%5 == 0:
3         print("FizzBuzz")
4     elif i%3 == 0:
5         print("Fizz")
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - If divisible by both 3 and 5, print "FizzBuzz".
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - Otherwise print the number.
  - Also should print a new line (so each entry is on its own line).

```
1 for i in range(1,101):
2     if i%3 == 0 and i%5 == 0:
3         print("FizzBuzz")
4     elif i%3 == 0:
5         print("Fizz")
6     elif i%5 == 0:
7         print("Buzz")
```

Tech Interview Classic

- To Do List:
  - Create a loop that goes from 1 to 100.
  - If divisible by both 3 and 5, print "FizzBuzz".
  - If the number is divisible by 3, print "Fizz".
  - If the number is divisible by 5, print "Buzz".
  - Otherwise print the number.
  - Also should print a new line (so each entry is on its own line).

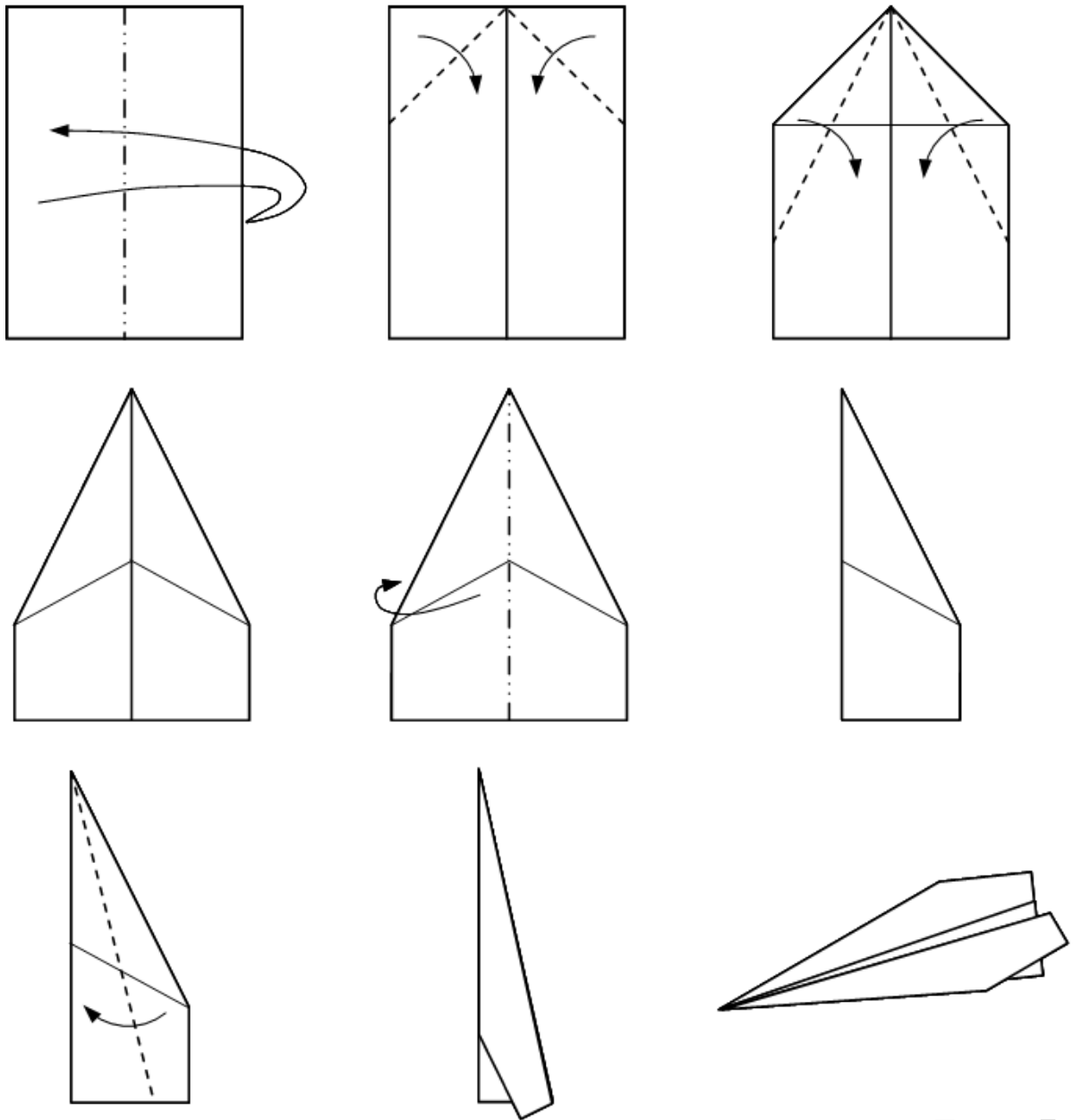
```
1 for i in range(1,101):
2     if i%3 == 0 and i%5 == 0:
3         print("FizzBuzz")
4     elif i%3 == 0:
5         print("Fizz")
6     elif i%5 == 0:
7         print("Buzz")
8     else:
9         print(i)
```

link.

Today's Topics

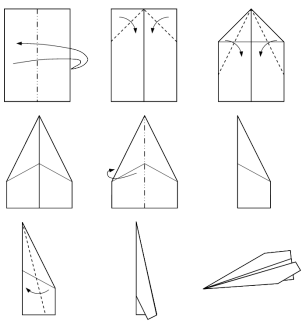
- Recap: Decisions
- Logical Expressions
- Circuits
- Binary Numbers
- **Design Challenge: Airplanes**

Design Challenge: Planes



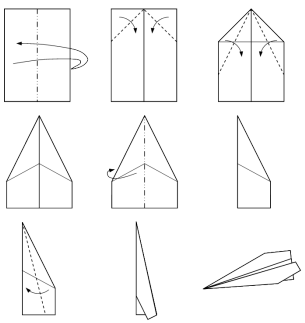
Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.



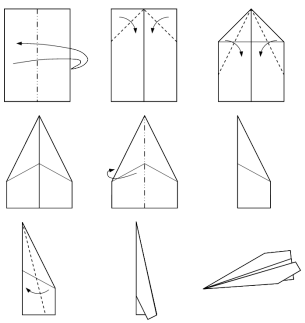
Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist:



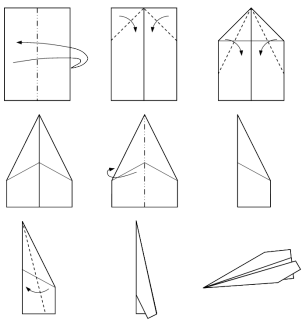
Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs



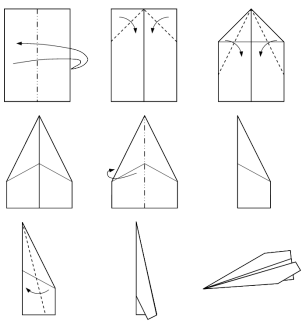
Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.

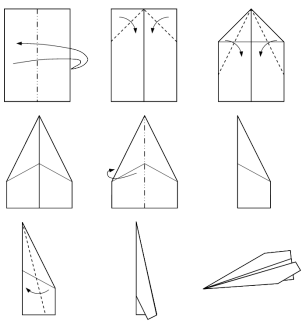


Design Challenge: Planes

- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.

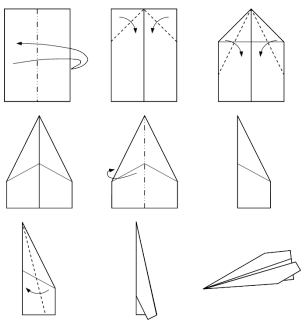


Design Challenge: Planes



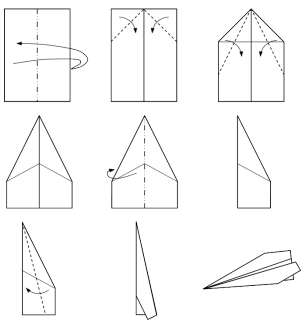
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) **without consulting you**.

Design Challenge: Planes



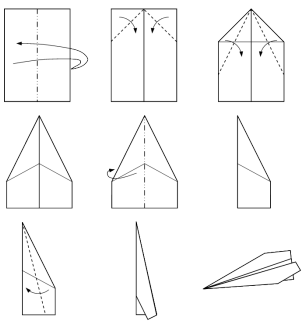
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) **without consulting you**.
  - You exchange test planes, and **revise your algorithm**.

Design Challenge: Planes



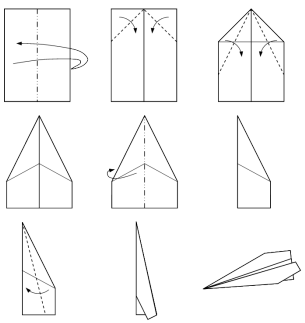
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) **without consulting you**.
  - You exchange test planes, and **revise your algorithm**.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT)

Design Challenge: Planes



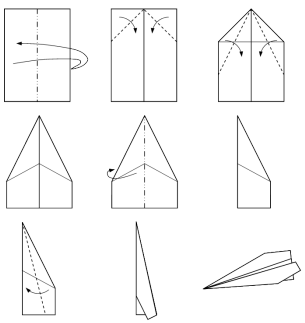
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) **without consulting you**.
  - You exchange test planes, and **revise your algorithm**.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).

Design Challenge: Planes



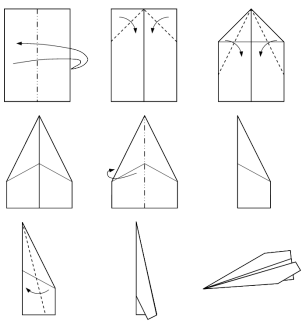
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) **without consulting you**.
  - You exchange test planes, and **revise your algorithm**.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.

Design Challenge: Planes



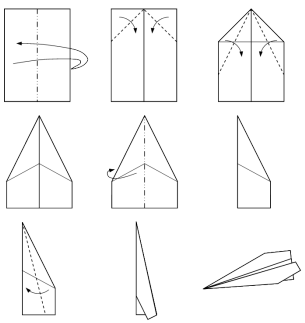
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) **without consulting you**.
  - You exchange test planes, and **revise your algorithm**.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.

Design Challenge: Planes



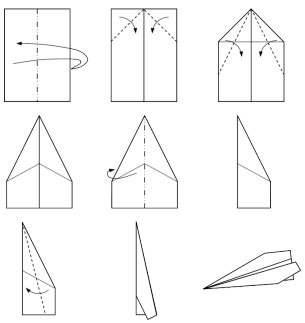
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) **without consulting you**.
  - You exchange test planes, and **revise your algorithm**.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.
- Remember to pick up all your airplanes!

Design Challenge: Initial Design **(2 Minutes)**



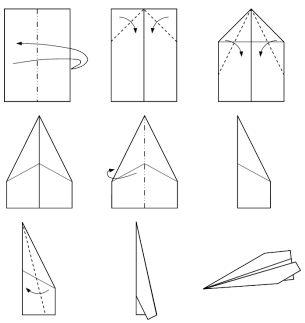
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - **As a team, write down your design.**
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) without consulting you.
  - You exchange test planes, and revise your algorithm.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.
- Remember to pick up all your airplanes!

Design Challenge: Test Build **(2 Minutes)**



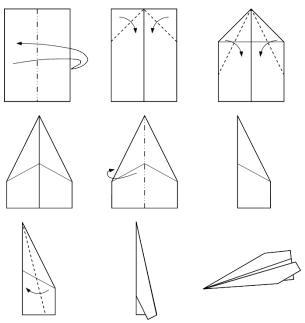
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - **Exchange with another team.**
  - **They build an airplane to your design (TEST FLIGHT) without consulting you.**
  - You exchange test planes, and revise your algorithm.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.
- Remember to pick up all your airplanes!

Design Challenge: Revise Design **(3 Minutes)**



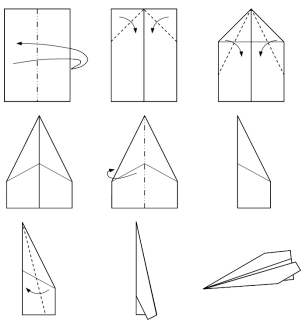
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) without consulting you.
  - **You exchange test planes, and revise your algorithm**.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.
- Remember to pick up all your airplanes!

Design Challenge: Build Final Planes **(2 Minutes)**



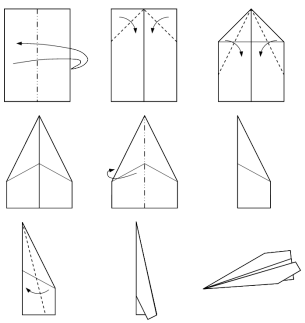
- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design ((TEST FLIGHT) without consulting you.
  - You exchange test planes, and revise your algorithm.
  - **The build team makes a copy of your revised paper airplane (FINAL FLIGHT)** and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.
- Remember to pick up all your airplanes!

Design Challenge: Test Planes **(3 Minutes)**



- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) without consulting you.
  - You exchange test planes, and revise your algorithm.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) **and flies it from the balcony (must be behind first row of seats)**.
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.
- Remember to pick up all your airplanes!

Design Challenge: Retrieve Planes **(2 Minutes)**



- A classic write-an-algorithm challenge for introductory programming.
- With a slight twist: refining designs
  - As a team, write down your design.
  - Exchange with another team.
  - They build an airplane to your design (TEST FLIGHT) without consulting you.
  - You exchange test planes, and revise your algorithm.
  - The build team makes a copy of your revised paper airplane (FINAL FLIGHT) and flies it from the balcony (must be behind first row of seats).
  - Will be judged on closeness to the stage.
  - Winning design/build team gets chocolate.
- **Remember to pick up all your airplanes!**

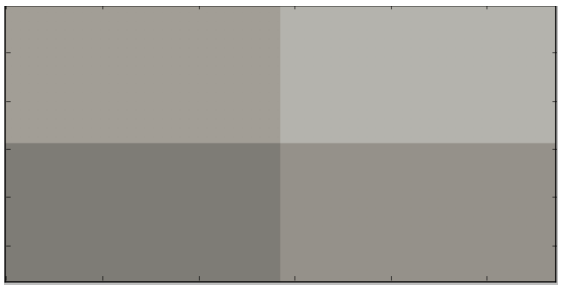
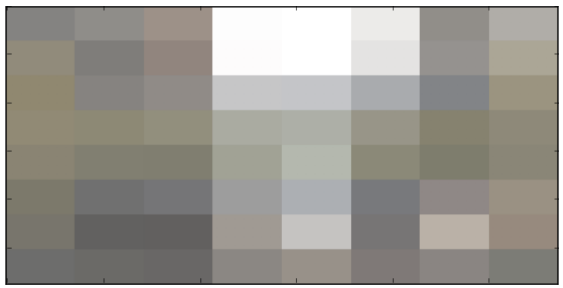
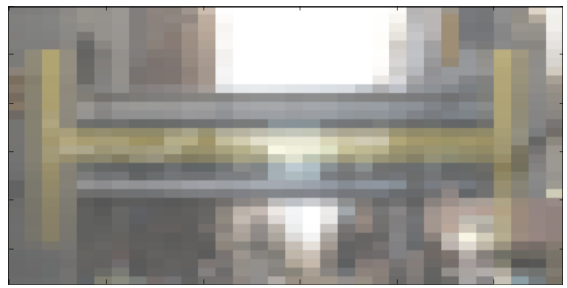
Recap

- In Python, we introduced:

Recap

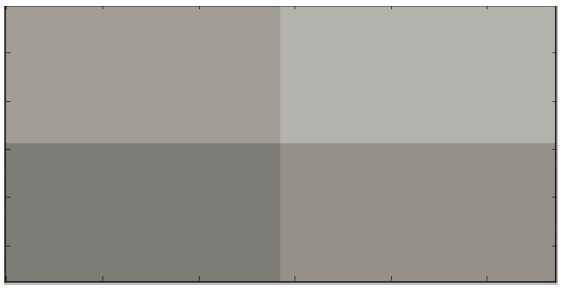
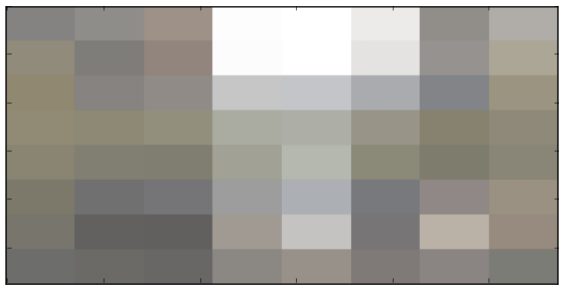
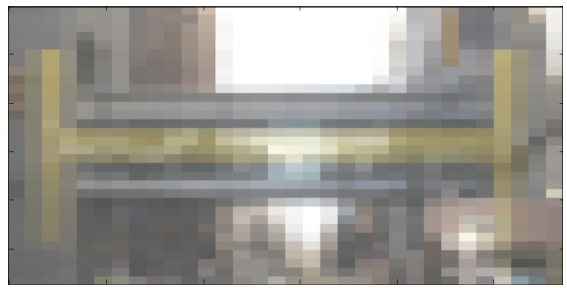
- In Python, we introduced:
  - Decisions
  - Logical Expressions
  - Circuits
  - Binary Numbers
  - Design Challenge: Airplanes

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:
  - 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.

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:

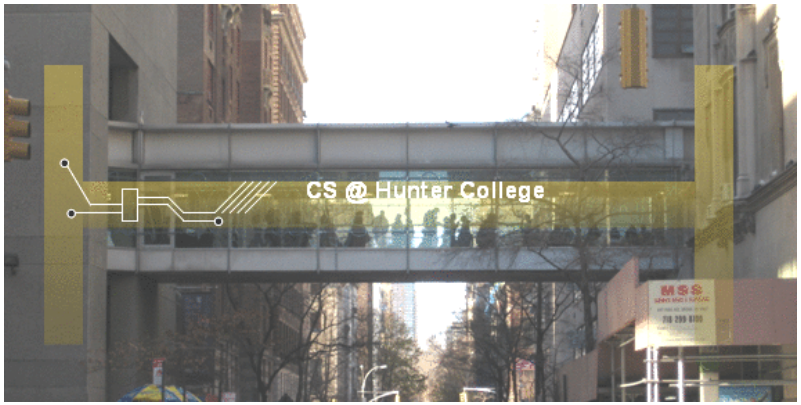
- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North

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 (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments **(programs 21-25)**

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 (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments **(programs 21-25)**
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5:30pm

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 (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments **(programs 21-25)**
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5:30pm
- 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.