You can access these slides on the course Github: https://github.com/natrask/ENM1050

ENGR 1050
Intro to Scientific Computation

Lecture 03 – Conditionals and user input

Prof. Nat Trask
Mechanical Engineering & Applied Mechanics
University of Pennsylvania

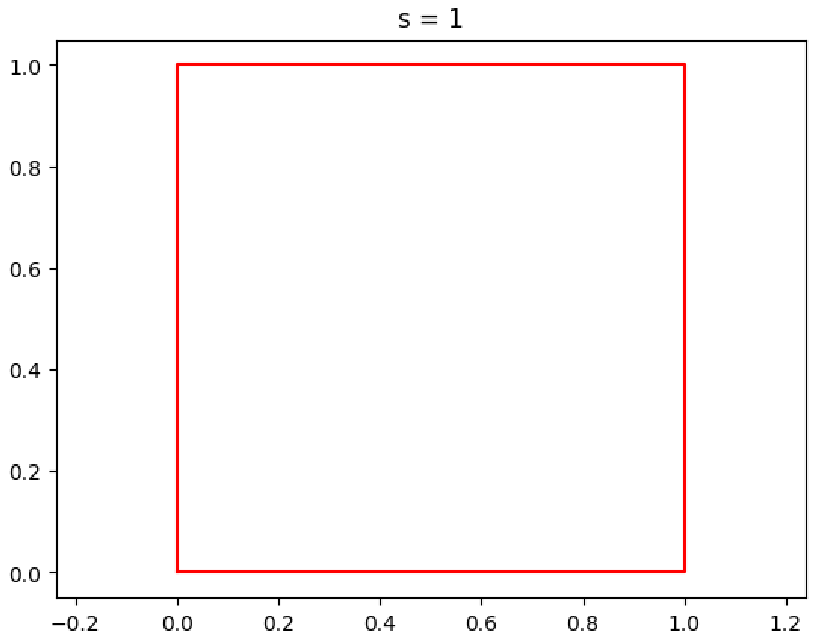
Review: What does this code output? Sketch it out.

```
import matplotlib.pyplot as plt

list1 = [0, 1, 1, 0, 0]
list2 = [0, 0, 1, 1, 0]

s = list1[1]-list2[1]

plt.plot(list1, list2, '-r')
plt.title("s = " + str(s))
plt.show()
```

It doesn’t quite look like a square. Try plt.axis("equal").

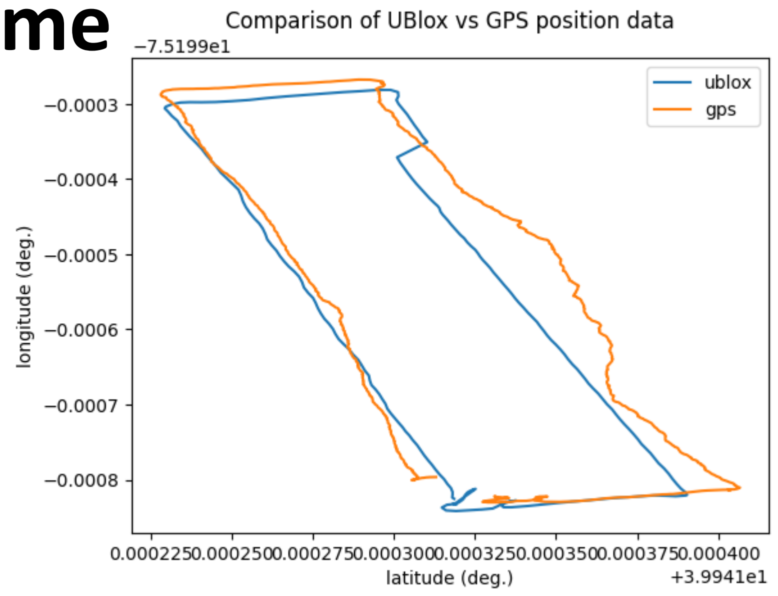
Review: What does this code output? Sketch it out.

```
import matplotlib.pyplot as plt

list1 = [0, 1, 1, 0, 0]
list2 = [0, 0, 1, 1, 0]

s = list1[1]-list2[1]

plt.plot(list1, list2, '-r')
plt.title("s = " + str(s))
plt.show()
```



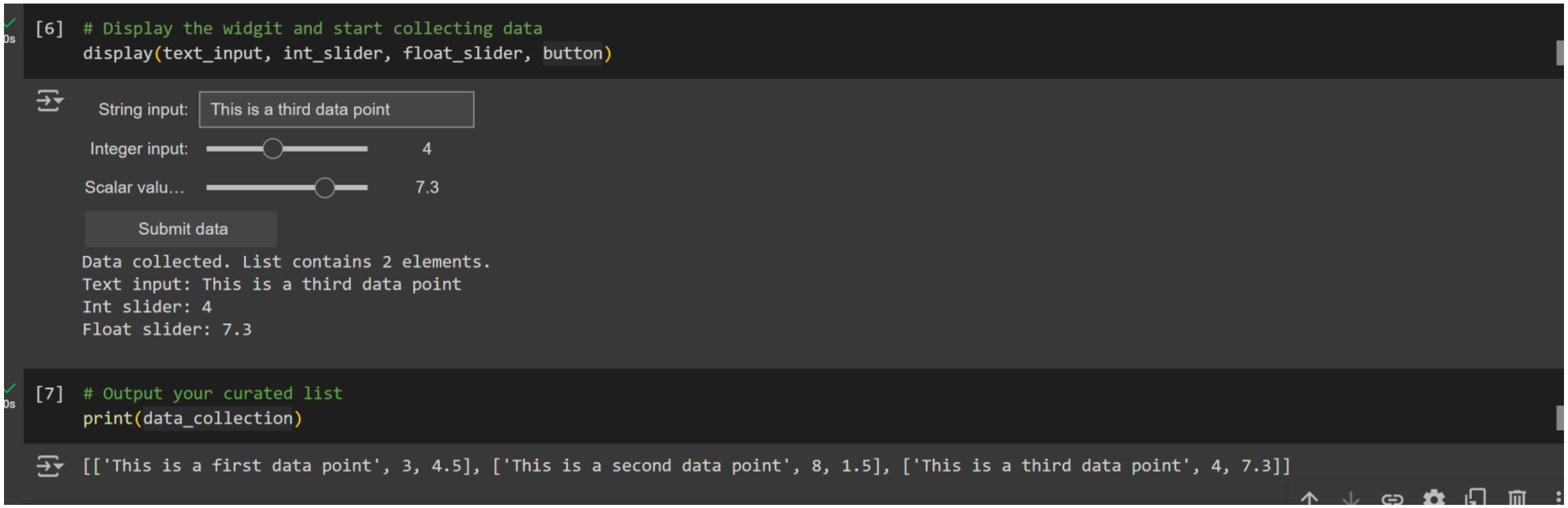
It doesn’t quite look like a square. Try plt.axis("equal").

Final plotting code from last time



```
# Final code from last time:

# Import packages for connecting to Spreadsheets
from google.colab import auth
auth.authenticate_user()

import gspread
from google.auth import default
creds, _ = default()

# Authorize Colab to access Spreadsheets
gc = gspread.authorize(creds)

# Import plotting packages
import matplotlib.pyplot as plt


## DATA
# data locations
ublox_filename = "https://docs.google.com/spreadsheets/d/1ahBDaVlCaEthfTmLysyGVbri69fKn5mYH8cQGETIotU/edit?usp=sharing"
gps_filename = "https://docs.google.com/spreadsheets/d/1H7acePHIh4ECncSFqkmaHNKdy6gnyFsOGZuK60xMjaY/edit?usp=sharing"

# load ublox data
worksheet = gc.open_by_url(ublox_filename).sheet1
data = worksheet.get_all_values()
latitude_ublox = convert_to_numbers(data[0])
longitude_ublox = convert_to_numbers(data[1])

# load gps data
worksheet = gc.open_by_url(gps_filename).sheet1
data = worksheet.get_all_values()
latitude_gps = convert_to_numbers(data[0])
longitude_gps = convert_to_numbers(data[1])


## PLOTTING
plt.plot(latitude_ublox, longitude_ublox, label="ublox")
plt.plot(latitude_gps, longitude_gps, label="gps")
plt.xlabel("latitude (deg.)")
plt.ylabel("longitude (deg.)")
plt.title("Comparison of UBlox vs GPS position data")
plt.legend()
plt.show()
```

Challenge code from last class:



```
[6] # Display the widgit and start collecting data
    display(text_input, int_slider, float_slider, button)
```

```
Data collected. List contains 2 elements.
Text input: This is a third data point
Int slider: 4
Float slider: 7.3
```

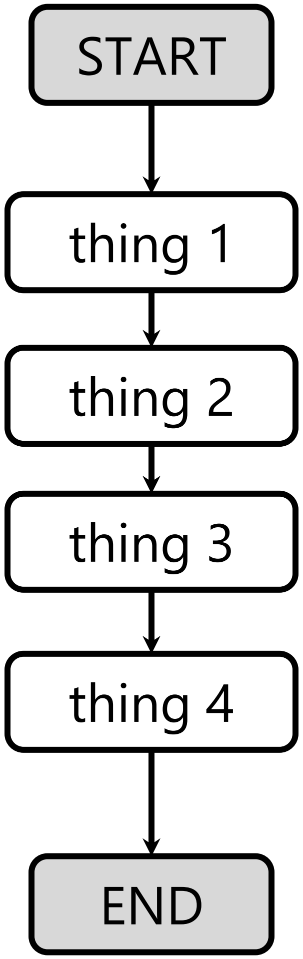
```
[7] # Output your curated list
    print(data_collection)
```

```
[['This is a first data point', 3, 4.5], ['This is a second data point', 8, 1.5], ['This is a third data point', 4, 7.3]]
```

https://github.com/natrask/ENM1050/blob/main/Code%20examples/Lecture04_datawidget.ipynb

Beyond sequential execution

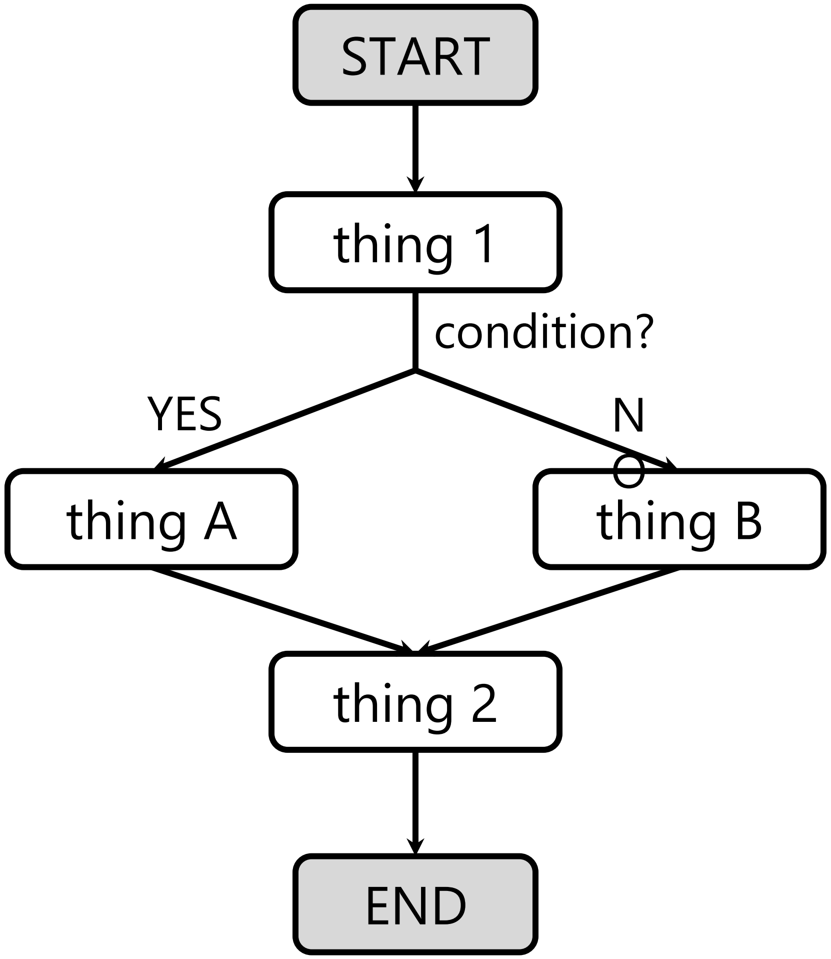
So far, our programs have looked like this



```
<do thing 1>

<do thing 2>

<do thing 3>

<do thing 4>
```

Beyond sequential execution

But … often, sequential execution is not enough



Control statements

Affect how other statements are executed

Conditionals: control which set of statements is executed

`if/else` statements   this week

Iterators: control how many times a set of statements is executed

`while` loops   next week

`for` loops

Boolean (logical) operators

Applied to Booleans, produces Booleans

x and y	true if both x and y are true
x or y	true if either x or y are true
not x	true if x is false

Review: Boolean types

True and False → False

True and True → True

False and False → True

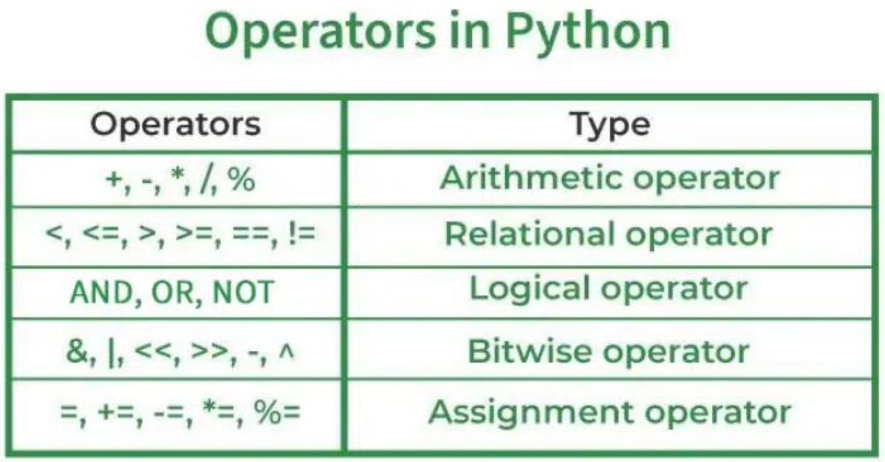
True or False → True

True or True → True

False or False → False

not False → True

not True → False

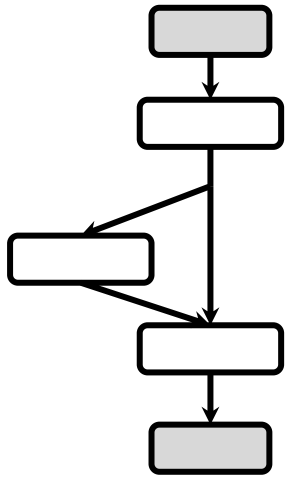


Operators in Python

Operators	Type
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
AND, OR, NOT	Logical operator
&, |, <<, >>, -, ^	Bitwise operator
=, +=, -=, *=, %=	Assignment operator

if statement

```
if CONDITION:
    BODY
```



CONDITION: any Boolean (true/false) statement

BODY: any set of statements

If the CONDITION is true, BODY gets executed

How do we write this part in python?

```
if p1 played rock and p2 played scissors
    player 1 wins!
…
```

Two flavors: **Boolean** and **Relational** operators

Relational operators

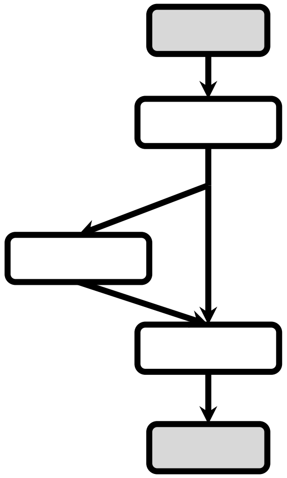
For comparing two variables

Applied to multiple types, always produces a **Boolean**

x == y	true if x equals y
	Don't confuse with assignment operator (x = y)!
x != y	true if x does not equal y
x > y	true if x greater than y
x < y	true if x less than y
x >= y	true if x greater than or equal to y
x <= y	true if x less than or equal to y

if statement

```
if CONDITION:
    BODY
```



CONDITION: any Boolean (true/false) statement

BODY: any set of statements

If the CONDITION is true, BODY gets executed

```
if p1 == "rock" and p2 == "scissors" :
    # player 1 wins!
...
```

Common errors

```
b = 1
if b = 0
    print(b)
   print("inside if maybe")
```

missing :

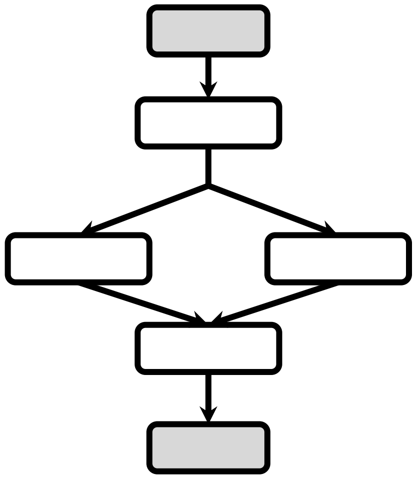
use ==, not =

SyntaxError: invalid syntax

IndentationError: unindent does not match any outer indentation level

if/else statement

```
if CONDITION:
    BODY1
else:
    BODY2
```



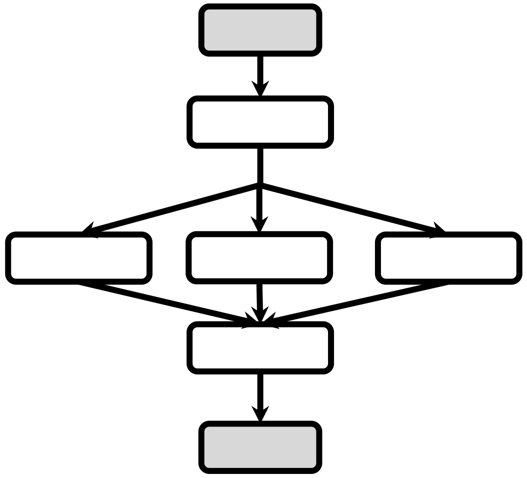
If the CONDITION is true, BODY1 gets executed

Otherwise, BODY2 gets executed

```
if p1 == "rock" and p2 == "scissors" :
    # player 1 wins!
else :
    # check other conditions
```

Chained conditionals

```
if CONDITION1:
    BODY1
elif CONDITION2:
    BODY2
else:
    BODY3
```



```
if p1 == "rock" :
    # do something
elif p1 == "paper" :
    # do something else
elif p1 == "scissors" :
    # do something ELSE
else :
    # player 1 played invalid hand
```

Order of if/elif/else blocks matter!

The order of conditional blocks makes a difference. This is because if the `if` or `elif` statement is `True`, the subsequent blocks are not executed.

```
if CONDITION1:
    BODY1
elif CONDITION2:
    BODY2
else:
    BODY3
```

Order of if/elif/else blocks matter!

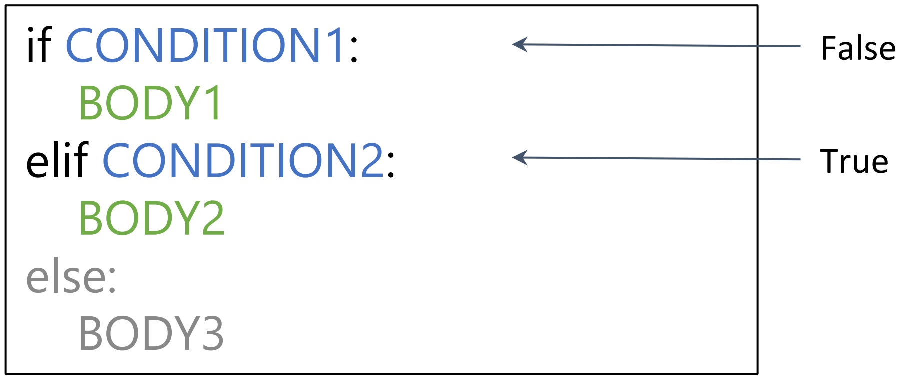
The order of conditional blocks makes a difference. This is because if the `if` or `elif` statement is `True`, the subsequent blocks are not executed.

```
if CONDITION1:
    BODY1
elif CONDITION2:
    BODY2
else:
    BODY3
```

True

Order of if/elif/else blocks matter!

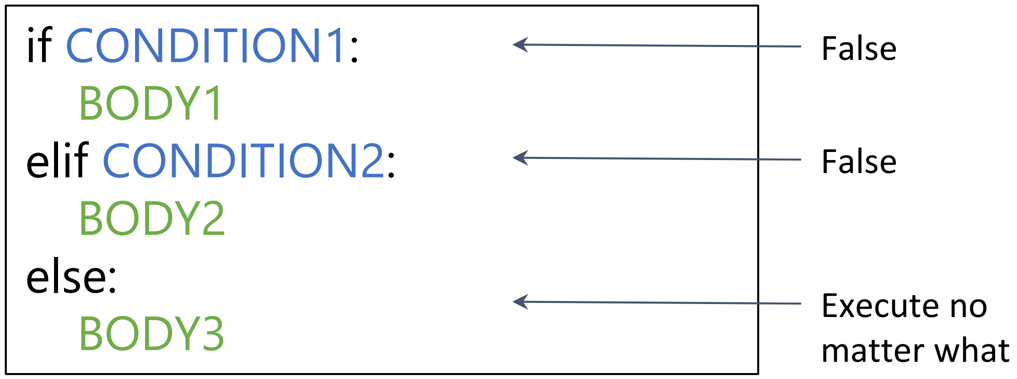
The order of conditional blocks makes a difference. This is because if the `if` or `elif` statement is `True`, the subsequent blocks are not executed.



```
if CONDITION1:
    BODY1
elif CONDITION2:
    BODY2
else:
    BODY3
```

Order of if/elif/else blocks matter!

The order of conditional blocks makes a difference. This is because if the `if` or `elif` statement is `True`, the subsequent blocks are not executed.



```
if CONDITION1:
    BODY1
elif CONDITION2:
    BODY2
else:
    BODY3
```

You'll see an example in the in-class today

Nested conditionals

```
if is_adult :
    if is_senior_citizen :
        print("Admission $2 off.")
    else:
        print("Full price.")
else :
    print("Admission $5 off.")
```

Indentation is keeps the code readable and is REQUIRED to keep the Python interpreter happy!

User input

An input is any information that is provided to the program. Here, we introduce keyboard inputs. This is one of many input types (keyboard, mouse, file, sensor). It is a neat way to interact with your program.

To require a user to provide an input before a code block continues running, you can write the following:

```
input('Current time: ')
```

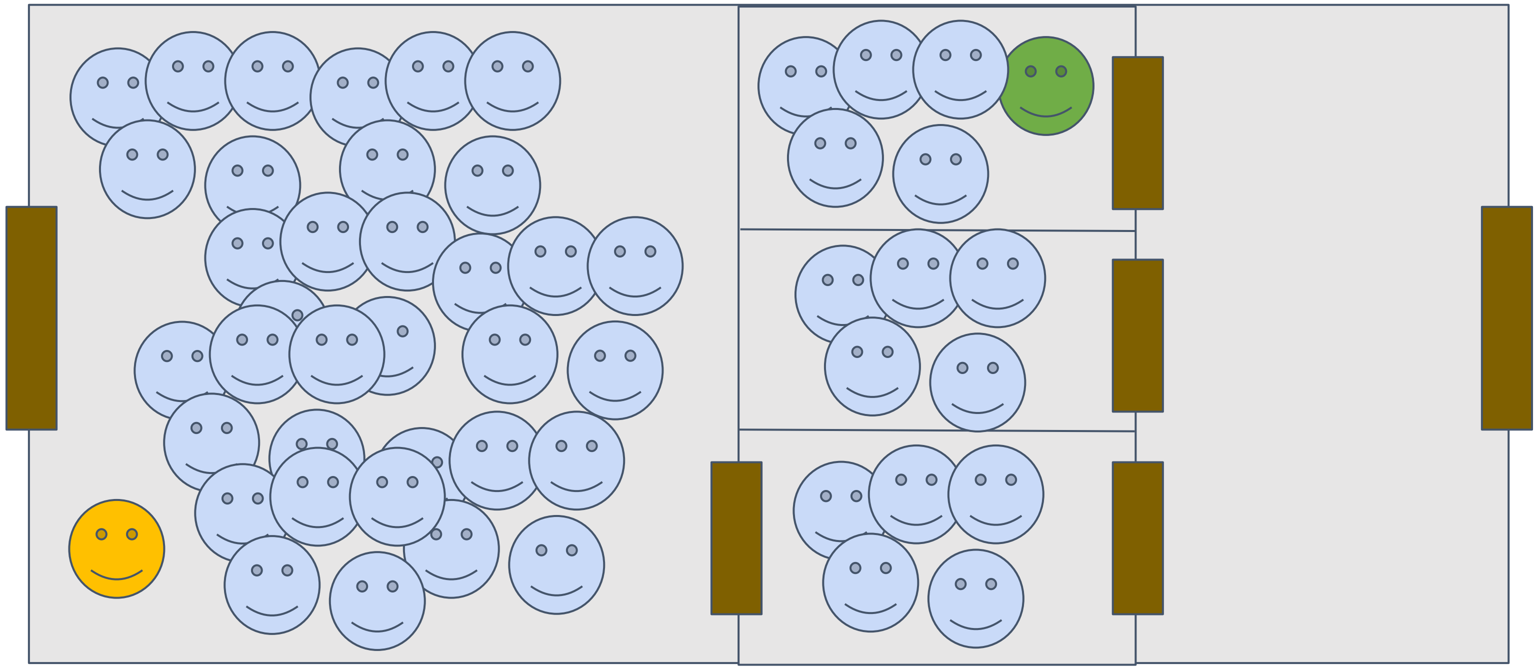
```
current_time = input('Current time: ')
```

$\leftarrow$ input is saved as a string

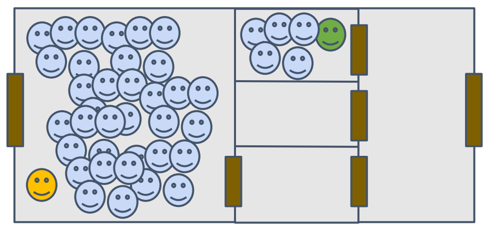
What navigation instructions to give?

You are at an event, and your friend calls trying to locate you. They have been told they are in the 'middle' of the building, what instructions do you give?

🟢	friend
🟡	you
🔵	others

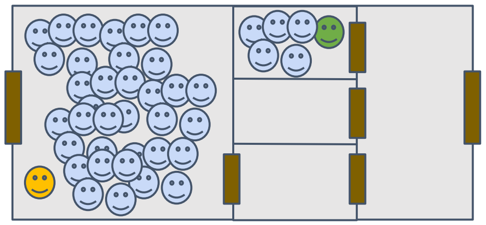


What navigation instructions to give?



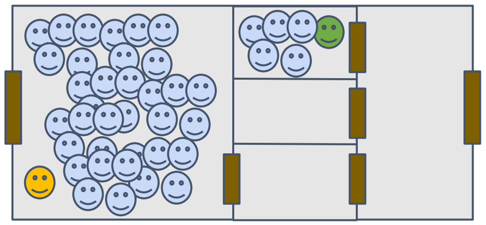
```
response = input('which door did you enter, left or right?')
if response == 'right':
        door = input('there are 3 doors there, are you in the 1st, 2nd, 3rd?')
        if door == '1st':
                print('then go through one more door, and walk straight ahead!)'
        elif door == '2nd':
                print('come out, go into the 1st')
                print('then go through one more door, and walk straight ahead!')
         elif door == '3rd':
                print('come out, go into the 1st')
                print('then go through one more door, and walk straight ahead!)
elif response == 'left':
                print('cool! I'm at the right of the main door once you walk in!')
```

What navigation instructions to give?



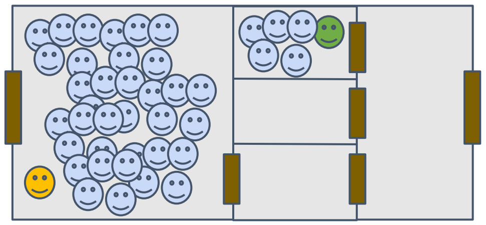
```
response = input(‘which door did you enter, left or right?’)
if response == ‘right’:
            door = input(‘there are 3 doors there, are you in the 1st, 2nd, 3rd?’)
            if door == ’1st’:
            print(‘then go through one more door, and walk straight ahead!)’
      elif door == ’2nd’:
                        print(‘come out, go into the 1st’)
            print(‘then go through one more door, and walk straight ahead!’)
      elif door == ‘3rd’:
            print(‘come out, go into the 1st’)
            print(‘then go through one more door, and walk straight ahead!)
elif response == ‘left’:
            print(‘cool! I’m at the right of the main door once you walk in!’)
```

What navigation instructions to give?



```
response = input('which door did you enter, left or right?')
if response == 'right':
    door = input('there are 3 doors there, are you in the 1st, 2nd, 3rd?')
    if door == '1st':
        print('then go through one more door, and walk straight ahead!)'
    elif door == '2nd':
        print('come out, go into the 1st')
        print('then go through one more door, and walk straight ahead!')
    elif door == '3rd':
        print('come out, go into the 1st')
        print('then go through one more door, and walk straight ahead!)
elif response == 'left':
        print('cool! I'm at the right of the main door once you walk in!')
```

What navigation instructions to give?



```
response = input('which door did you enter, left or right?')
if response == 'right':
    door = input('there are 3 doors there, are you in the 1st, 2nd, 3rd?')
    if door == '2nd' or door == '3rd':
        print('come out, go into the 1st')
        # note that 1st door does not need additional instructions
    print('then go through one more door, and walk straight ahead!')
else:
    print('cool! I'm at the right of the main door once you walk in')
```

inputs
nested conditionals
readability

In-Class: 03_Conditionals

Do this with a **different partner than last time**.

Turn in as a pair on Canvas.

Tips for pair programming:

- Switch off who is typing.
- The person who is not typing should:
  - Make comments or suggest potential solutions
  - Be "devil's advocate": what are potential issues with what is being typed
  - Suggest other things to explore

At-Home: HW 1 due Monday 11:59pm