k Nearest Neighbors

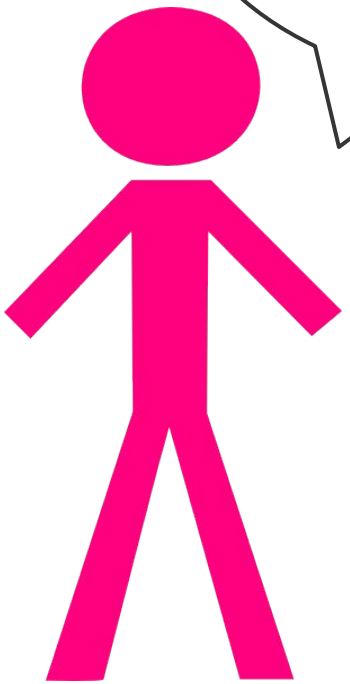
A brief introduction to a machine learning technique

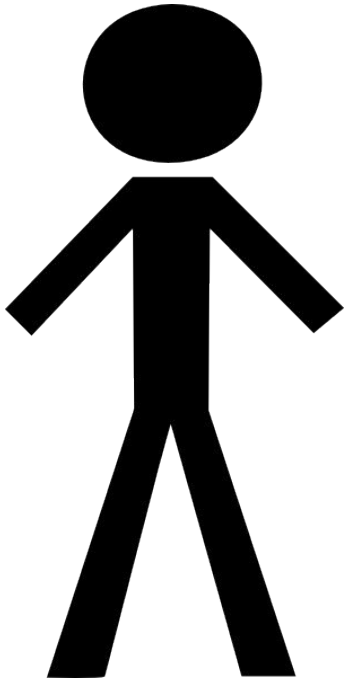
We Get Better At Stuff With Practice

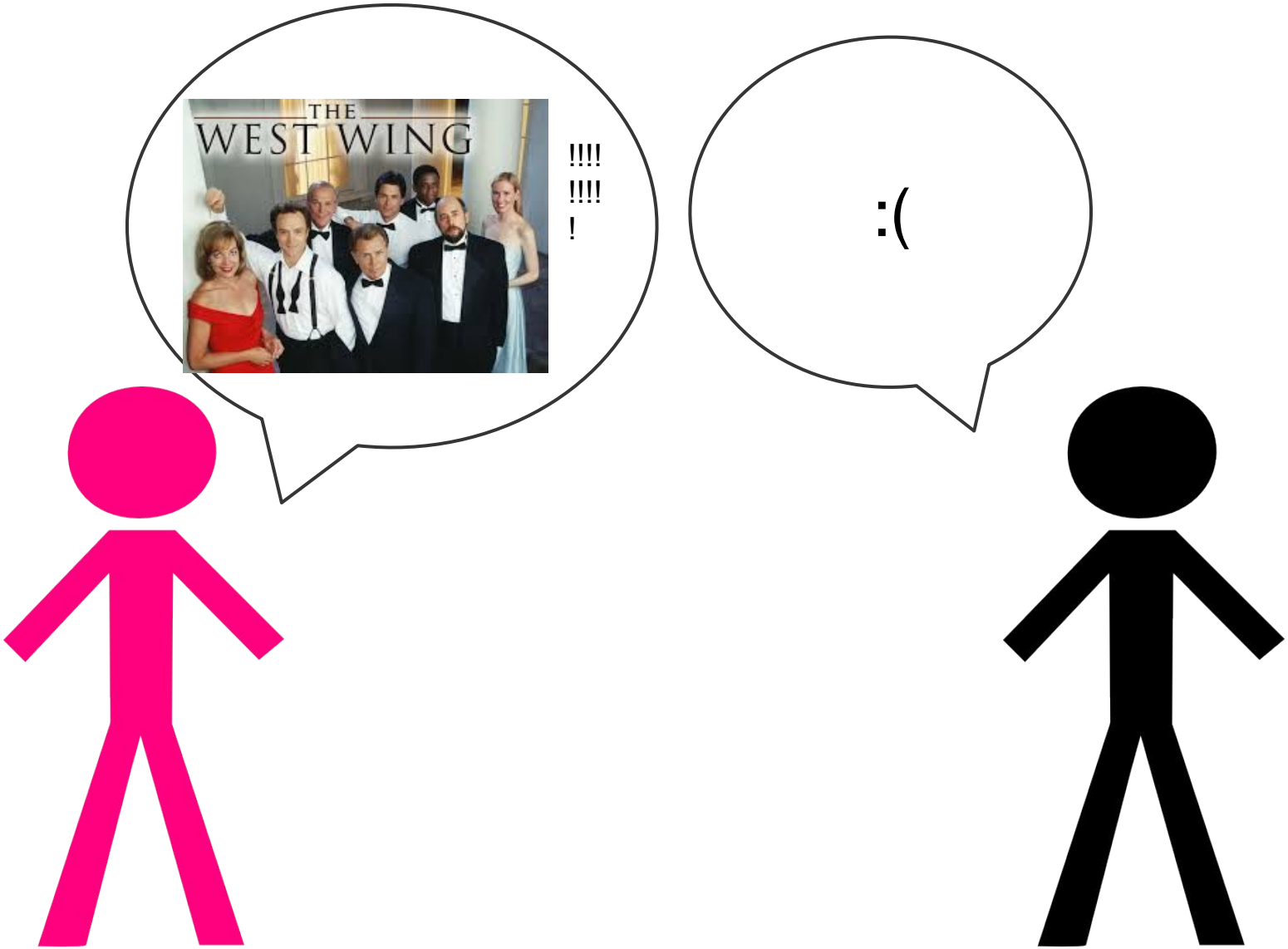
We Get Better At Stuff With Practice

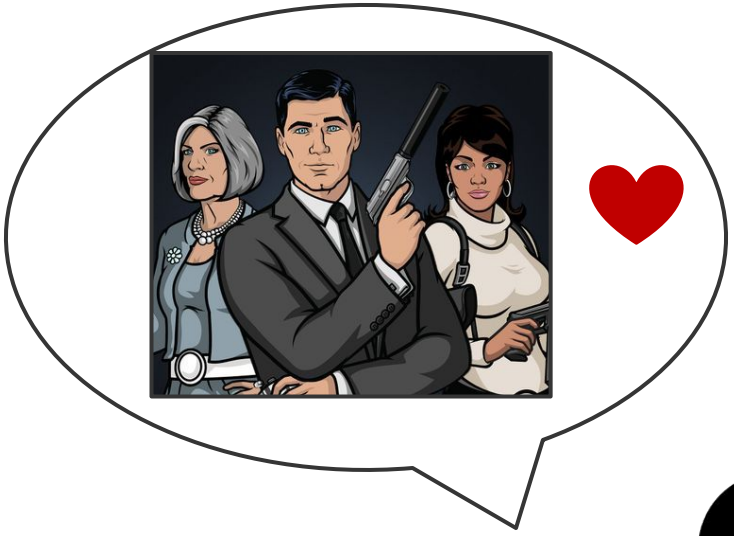
(like making recommendations)

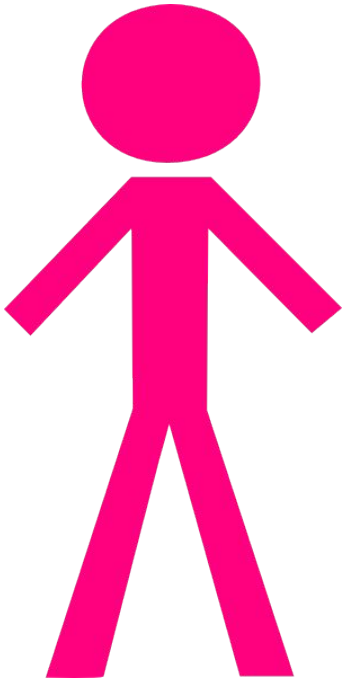

THE
WEST WING
!!!!
!!!!
!

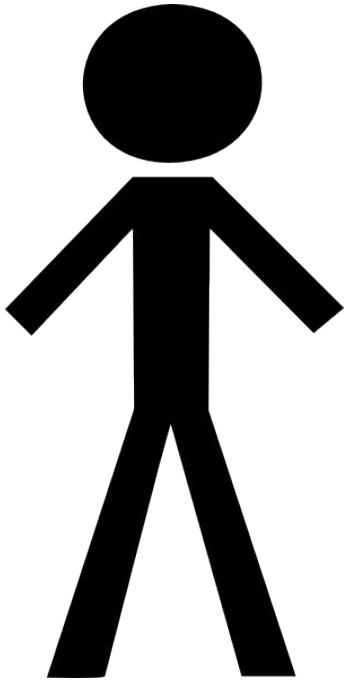


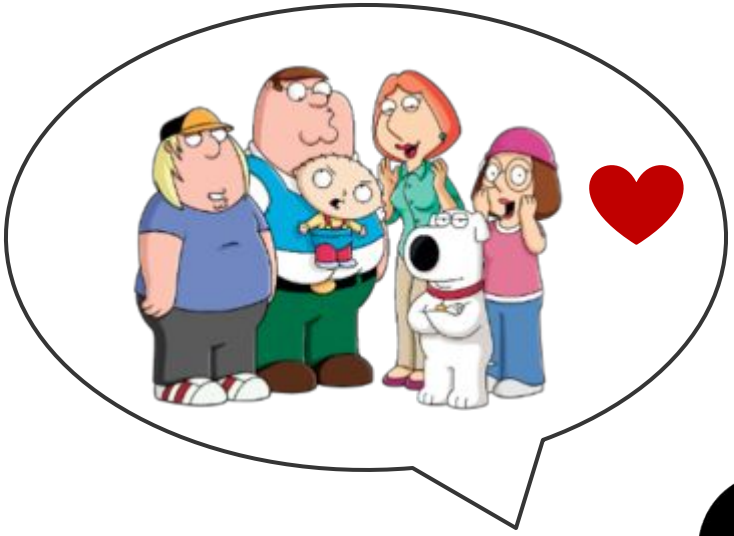


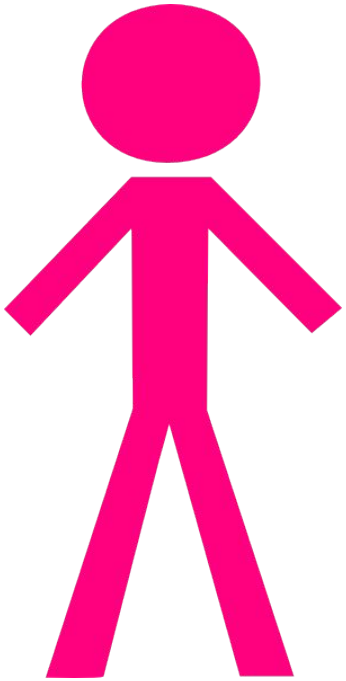

THE
WEST WING
!!!!
!!!!
!
:(

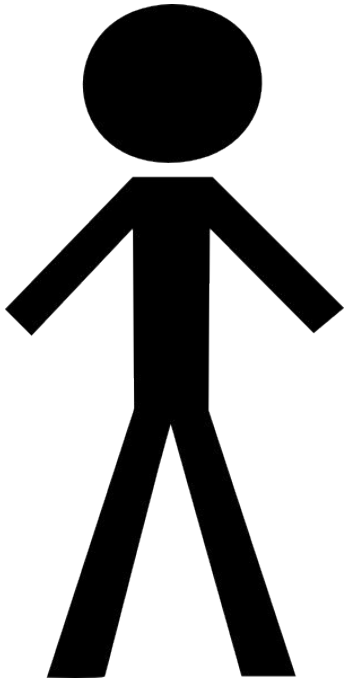


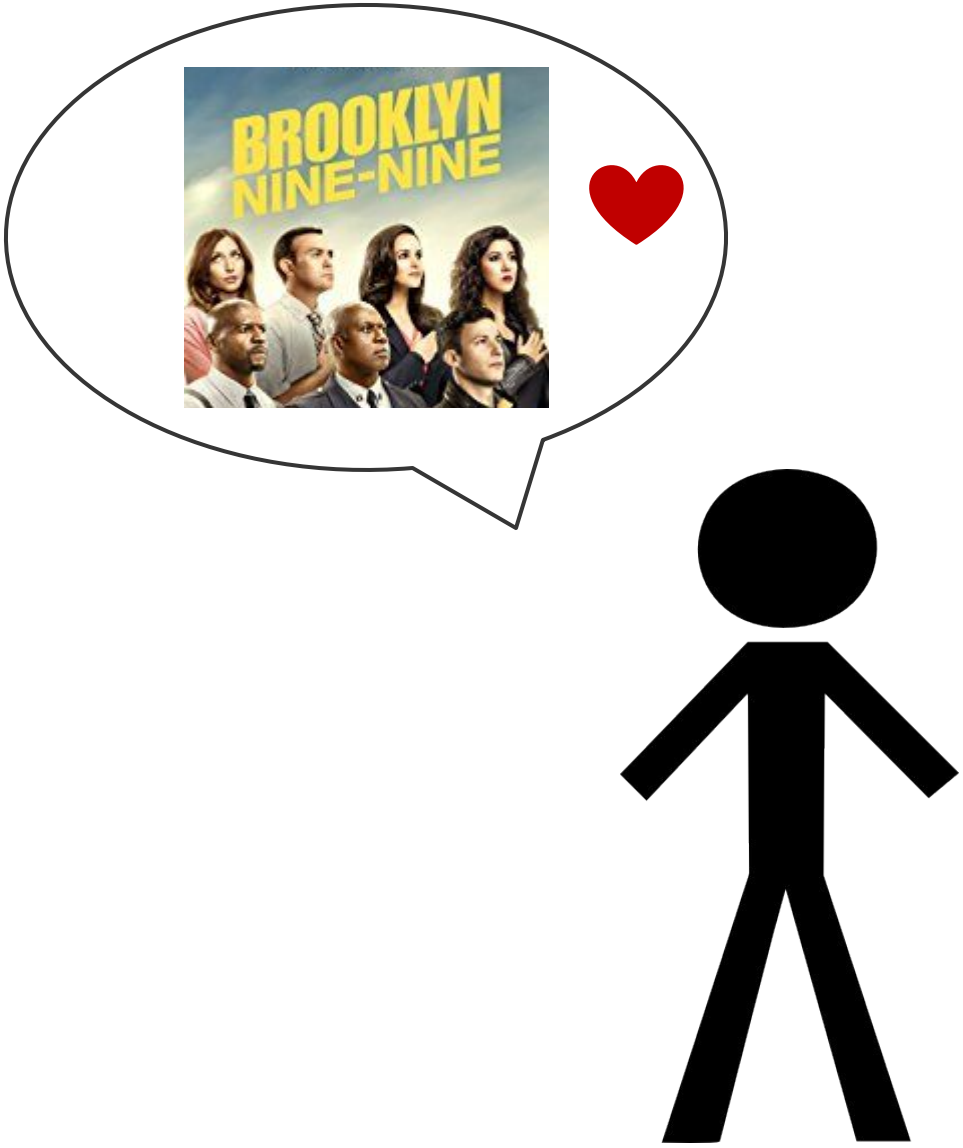


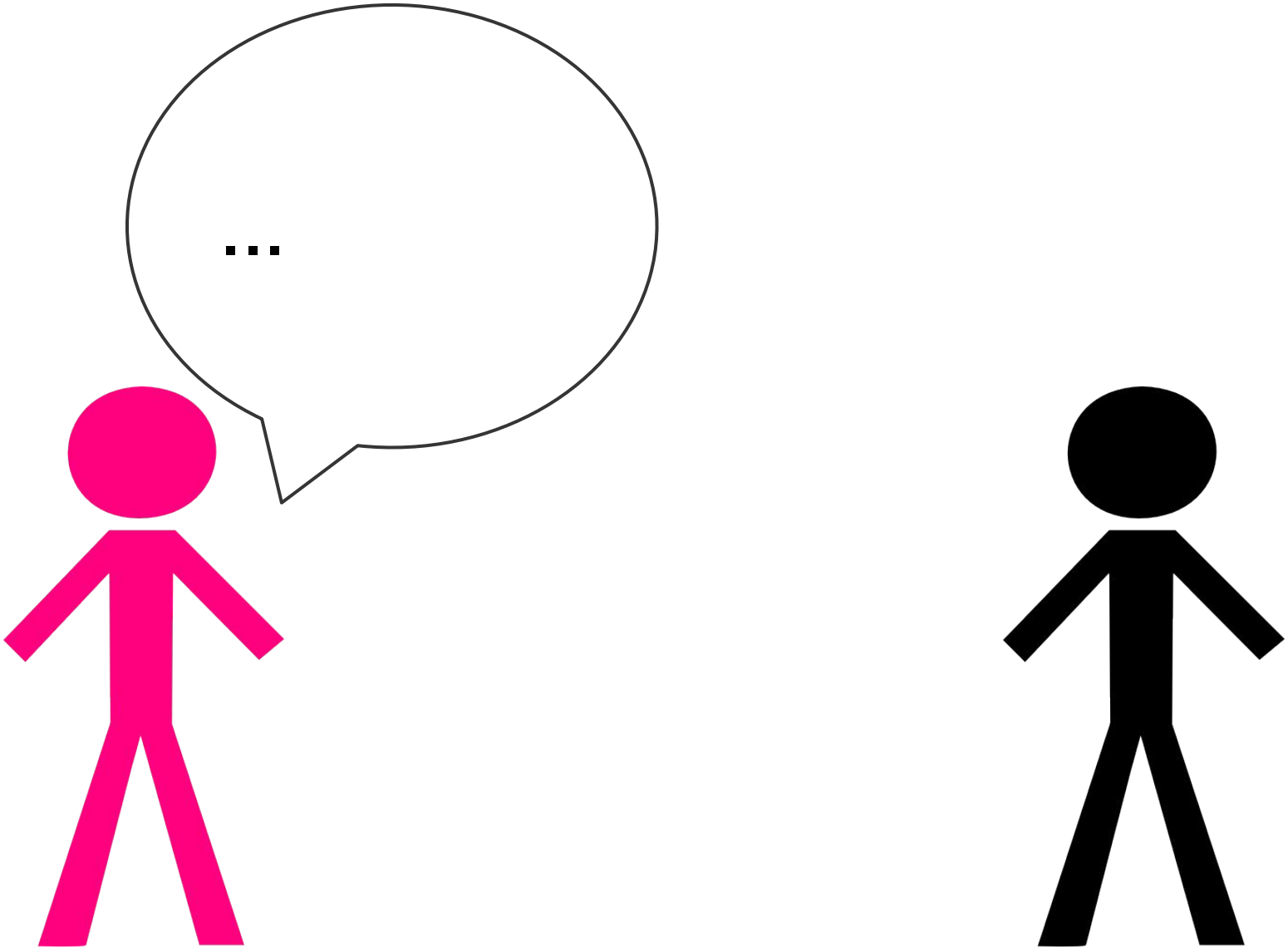


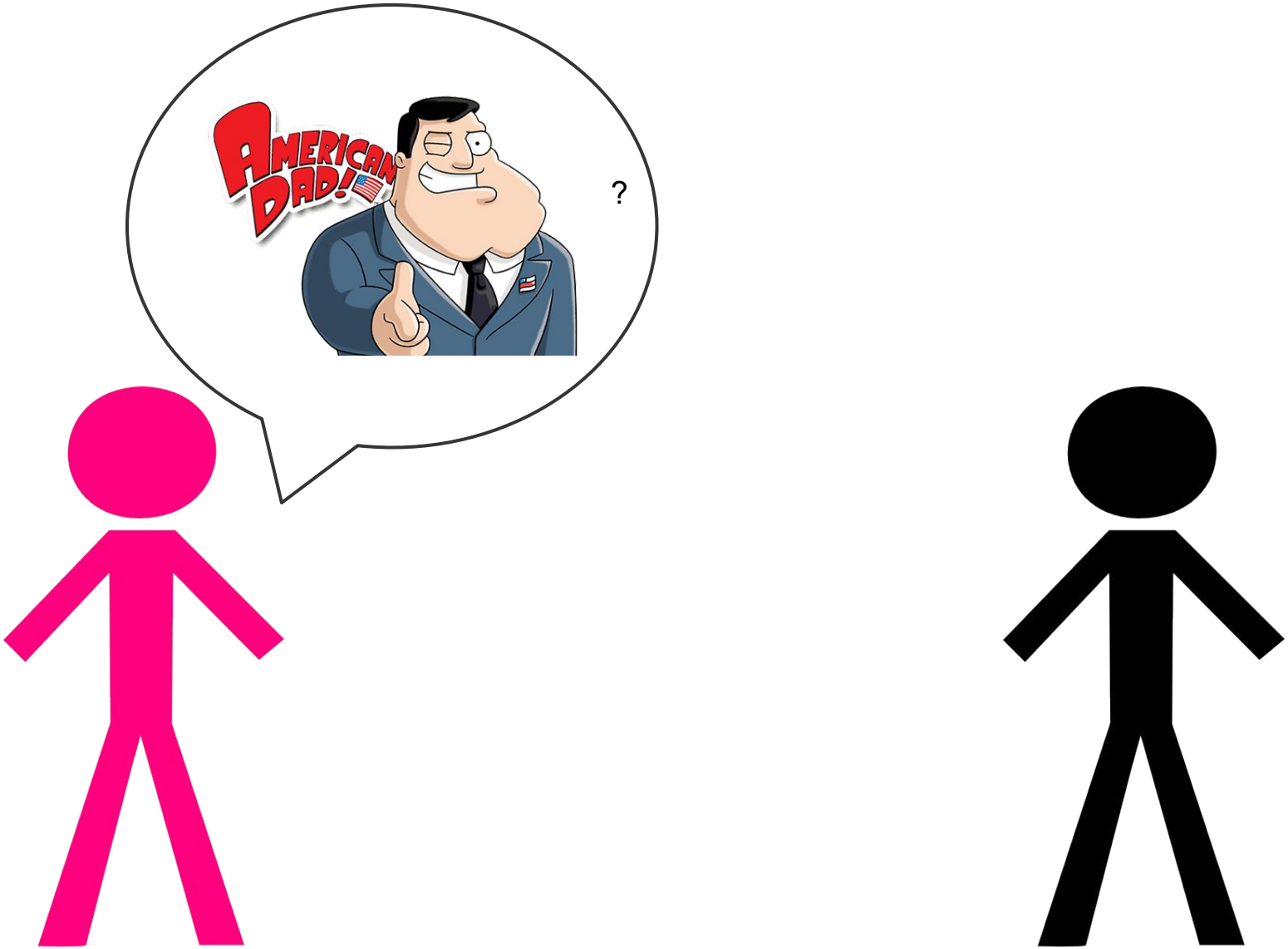


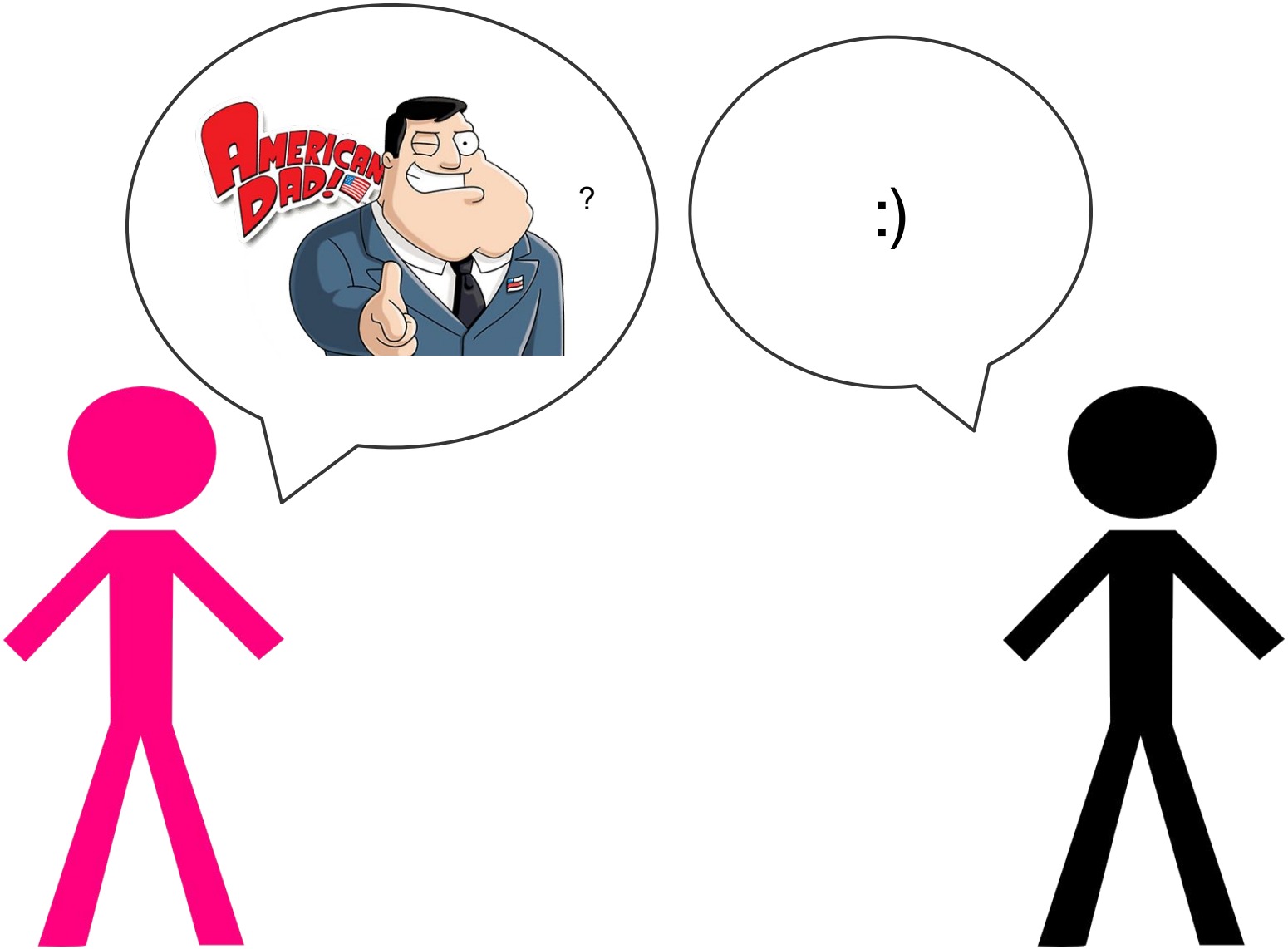





BROOKLYN
NINE-NINE

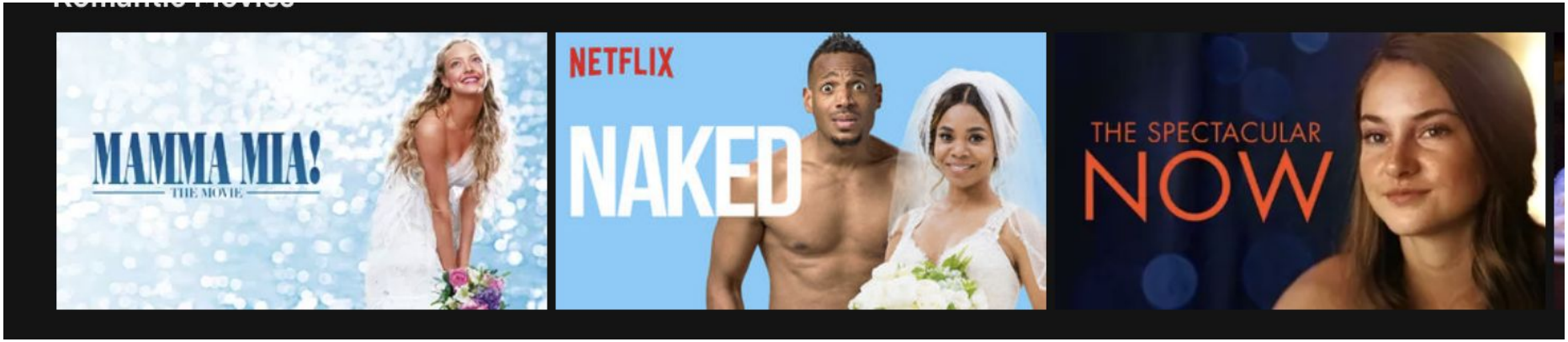



AMERICAN DAD!
?


AMERICAN DAD!
?
:)

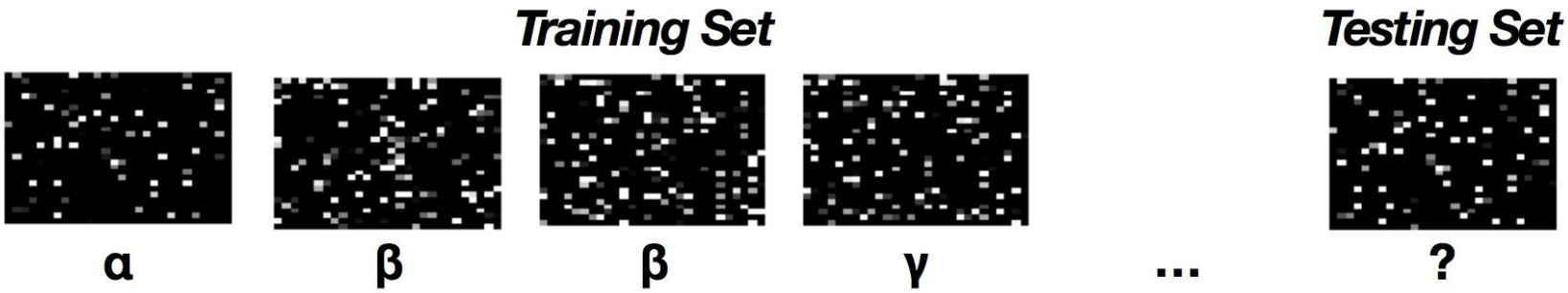
Computers Can Get Better With Practice, Too

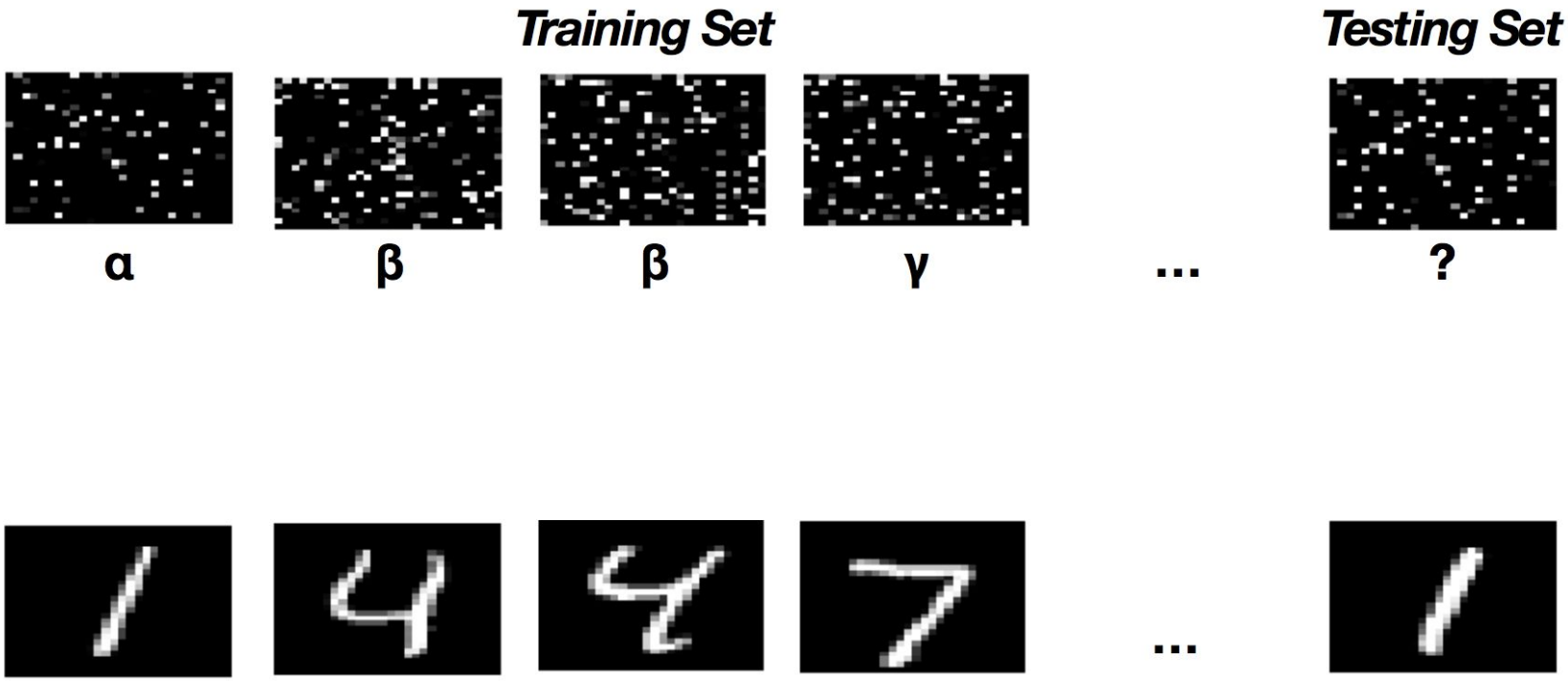
New User on Netflix



My Profile on Netflix

Machine learning: computers getting better with practice


Training Set
Testing Set
α
β
β
γ
...
?


Training Set
Testing Set
α
β
β
γ
...
?
...

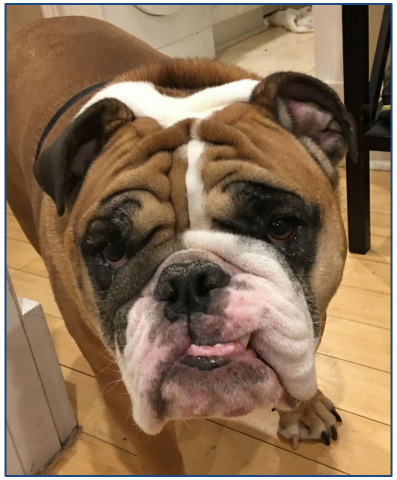
What is ***k Nearest Neighbors***?

What is ***k Nearest Neighbors***?

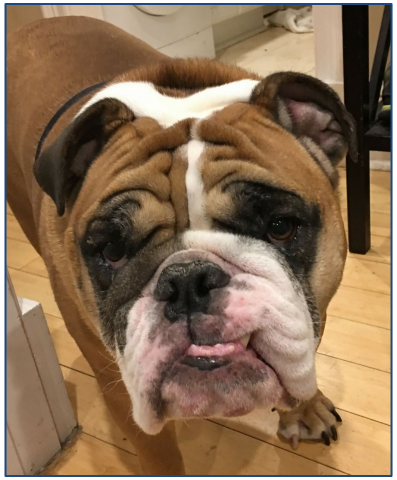
One Machine Learning technique!

Nearest Neighbor

- Input Data
  - A bunch of known information
- New Data
  - Find the neighbors nearest to input
  - Vote on output



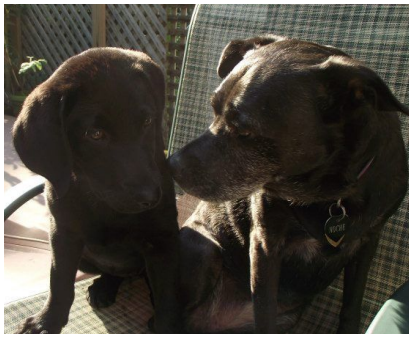
my dog :)

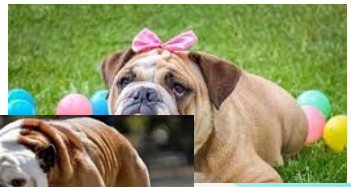


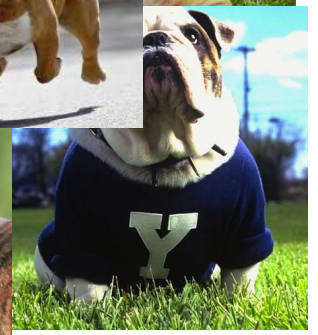
my dog :)

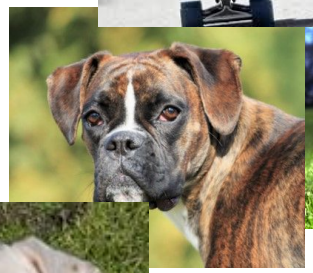
Where does Tugboat “fit”?

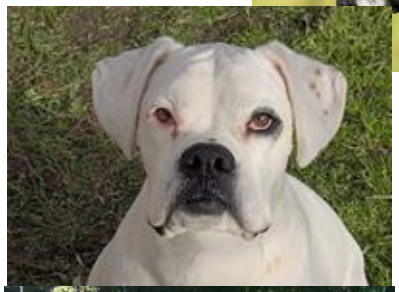

a bunch of
cute dogs

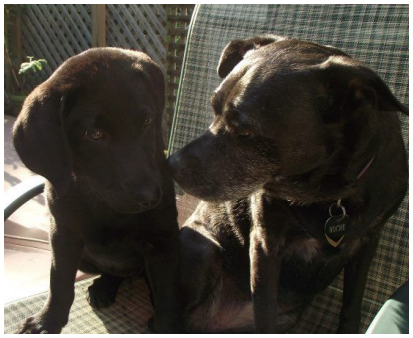


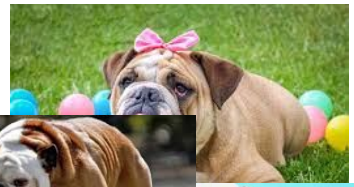


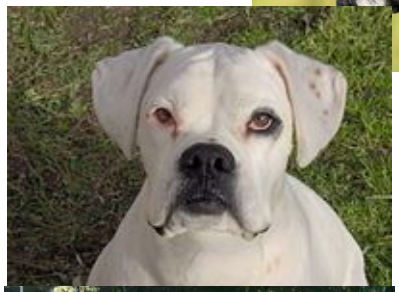


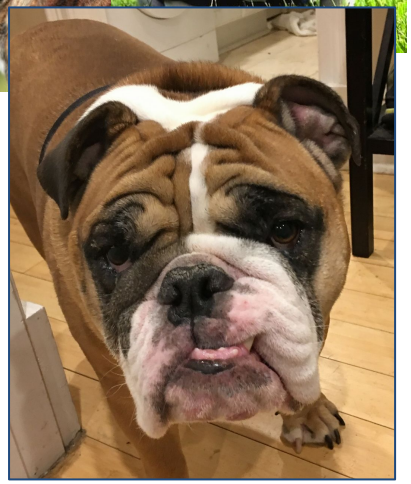












How Did We Decide?

- Length of Tail
- Color
- Sex
- Energy Level
- Size of Head
- Short/Long Hair
- Allergies
- Plays Fetch
- Likes Other Dogs
- Purebred/Mixed
- Age
- Weight
- Height
- Rescue Dog
- Likes Food
- Likes People
- Goes Swimming
- Quiet/Loud

How Did We Decide?

- Length of Tail
- Color
- Sex
- Energy Level
- Size of Head
- Short/Long Hair
- Allergies
- Plays Fetch
- Likes Other Dogs
- Purebred/Mixed
- Age
- Weight
- Height
- Rescue Dog
- Likes Food
- Likes People
- Goes Swimming
- Quiet/Loud

How We Use It

- What should I watch?
  - Given what I've already watched
- What is this restaurant like?
  - Given other restaurants nearby
- Whose face is this?
  - Given faces in my database
- What is the correct spelling?
  - Given similar misspelled words

There's less **distance** between neighbors

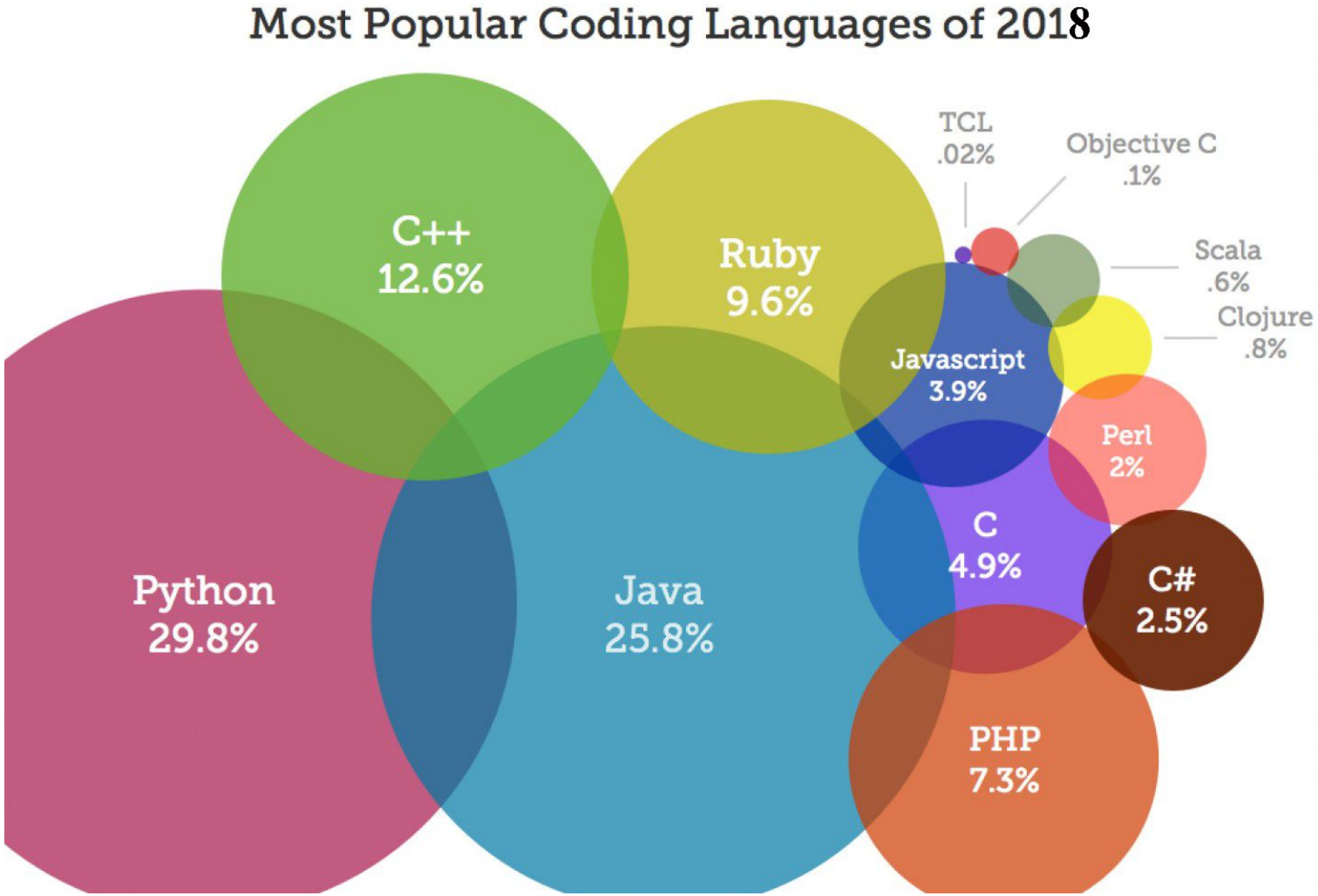
How do we measure the distance between two things?

How do we measure the distance between two things?

How do we measure the distance between two things?

There are lots of ways! **We'll implement 3 common ones.**

Implementation: in Python



https://www.dotnetlanguages.net/

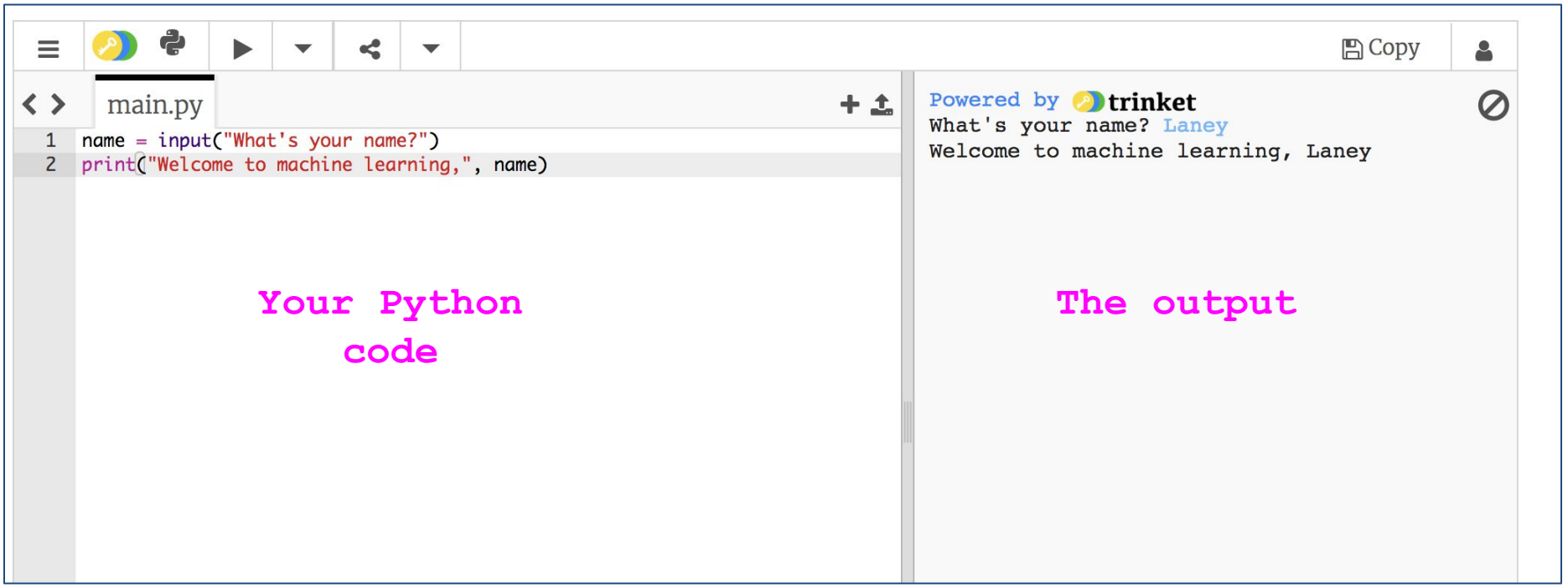
Python Is Used In...

- Machine Learning algorithms
- Web development
- Data analysis & visualization
- Bioinformatics labs
- Throwing together a quick script

https://trinket.io/python3

womenscoc

WomensCOC


Copy
main.py
1 name = input("What's your name?")
2 print("Welcome to machine learning,", name)
Your Python code
Powered by trinket
What's your name? Laney
Welcome to machine learning, Laney
The output

Warm up

Type this in your code:

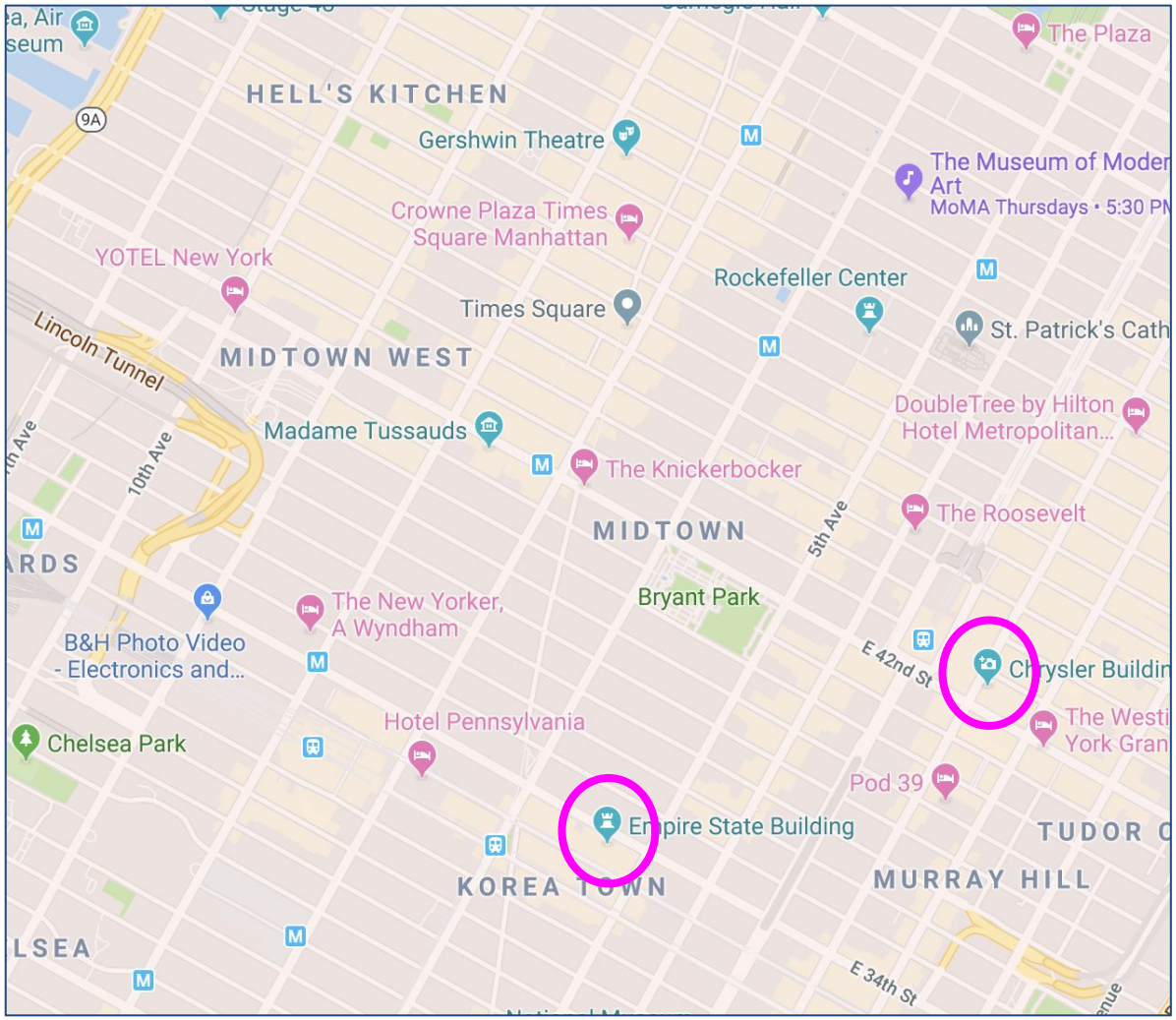
```
name = input("What's your name?")
print("Welcome to machine learning,", name)
```

Then click the run button: 

Warm-up over...

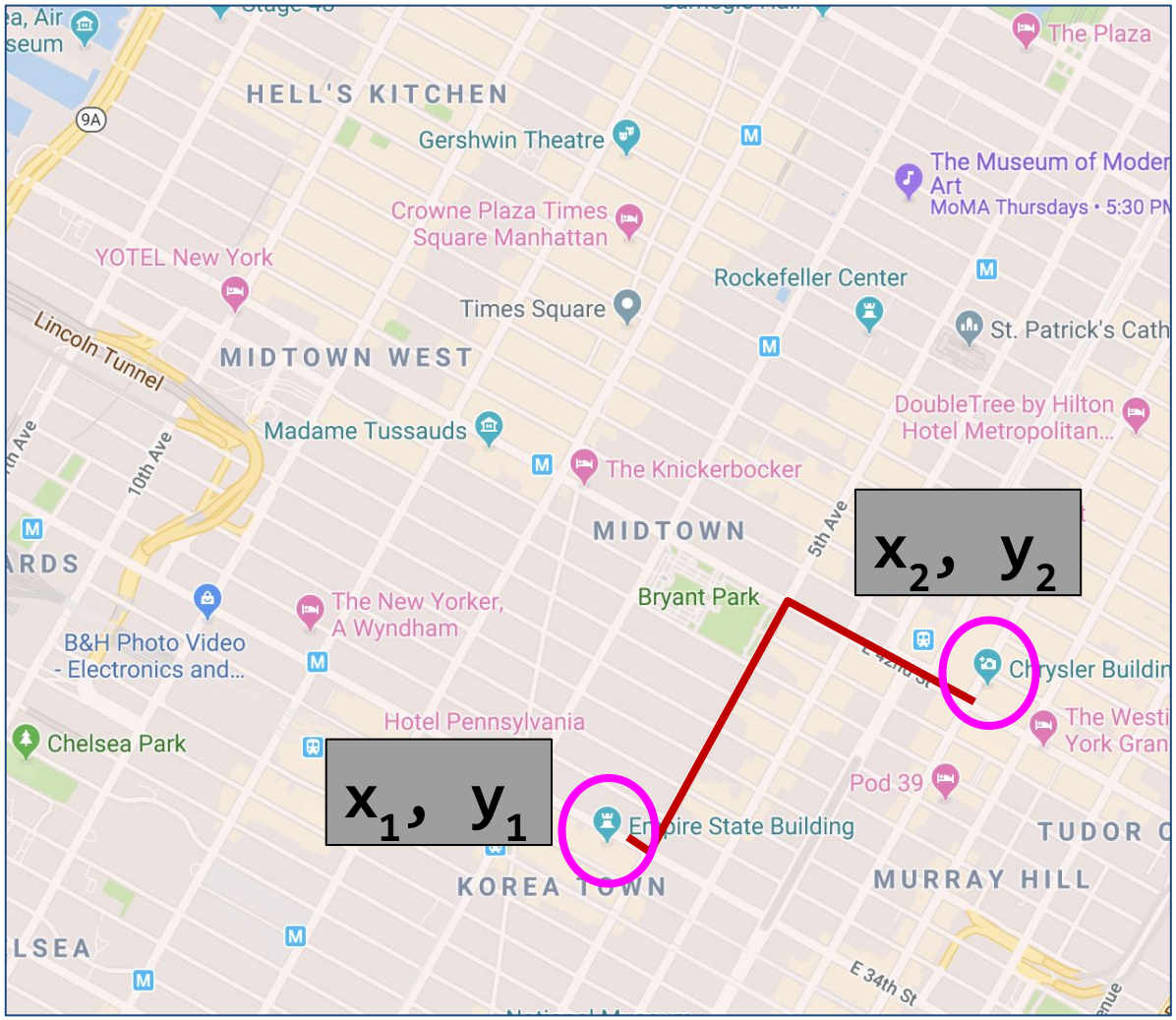
Now on to our distance functions

1. Manhattan Distance

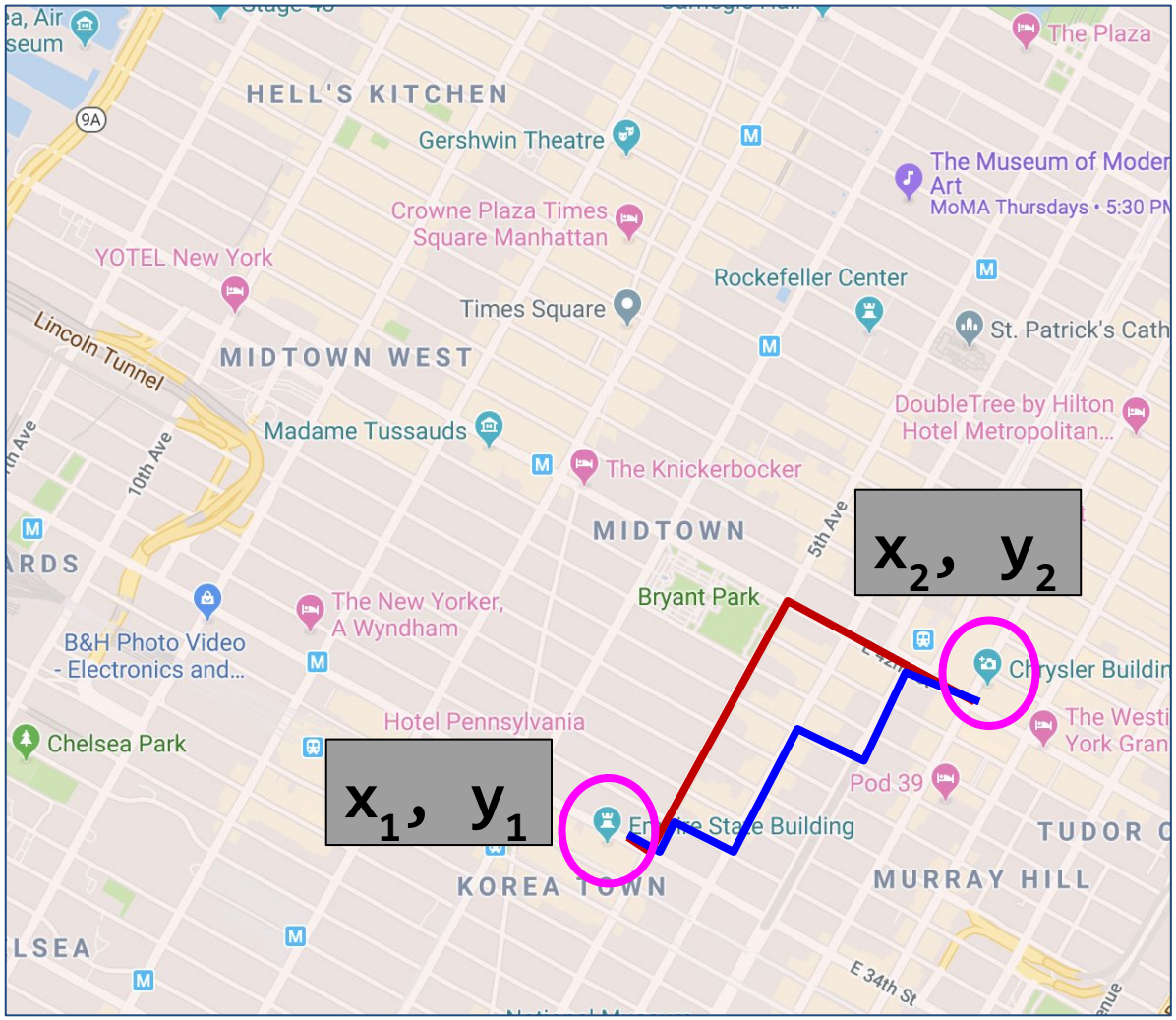


1. Manhattan Distance

1. Manhattan Distance



1. Manhattan Distance



1. Manhattan Distance

$$\text{Distance} = (x_2 - x_1) + (y_2 - y_1)$$

1. Manhattan Distance

$$\text{Distance} = (x_2 - x_1) + (y_2 - y_1)$$

```
def manhattan(x1, y1, x2, y2):
    x_difference = abs(x2 - x1)
    y_difference = abs(y2 - y1)
    return x_difference + y_difference
```

1. Manhattan Distance

How did we do?

```
def manhattan(x1, y1, x2, y2):
    x_difference = abs(x2 - x1)
    y_difference = abs(y2 - y1)
    return x_difference + y_difference

distance = manhattan(0, 0, 3, 4)
print(distance)
```

1. Manhattan Distance

How did we do?

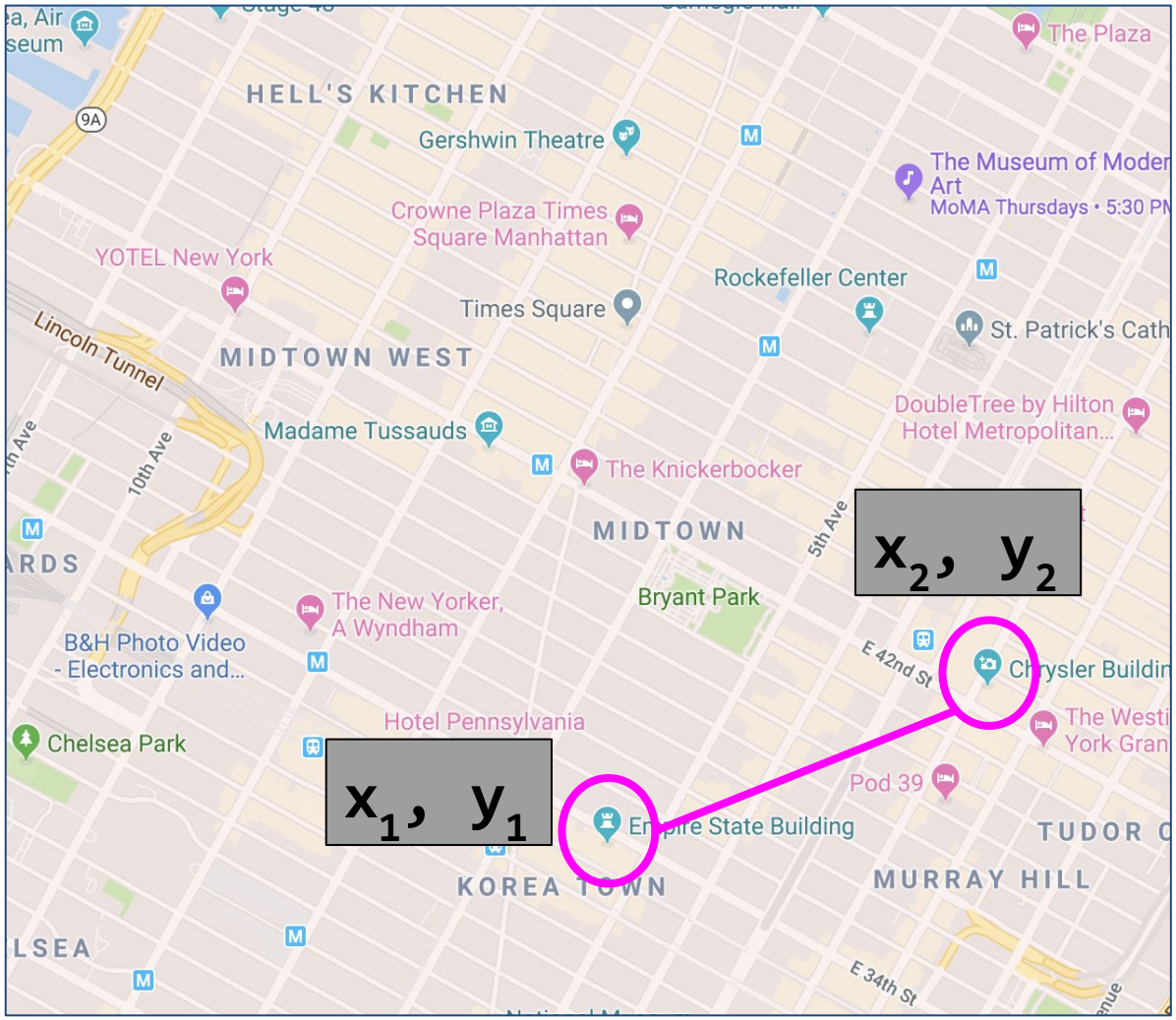
```
def manhattan(x1, y1, x2, y2):
    x_difference = abs(x2 - x1)
    y_difference = abs(y2 - y1)
    return x_difference + y_difference

distance = manhattan(0, 0, 3, 4)
print(distance)
```

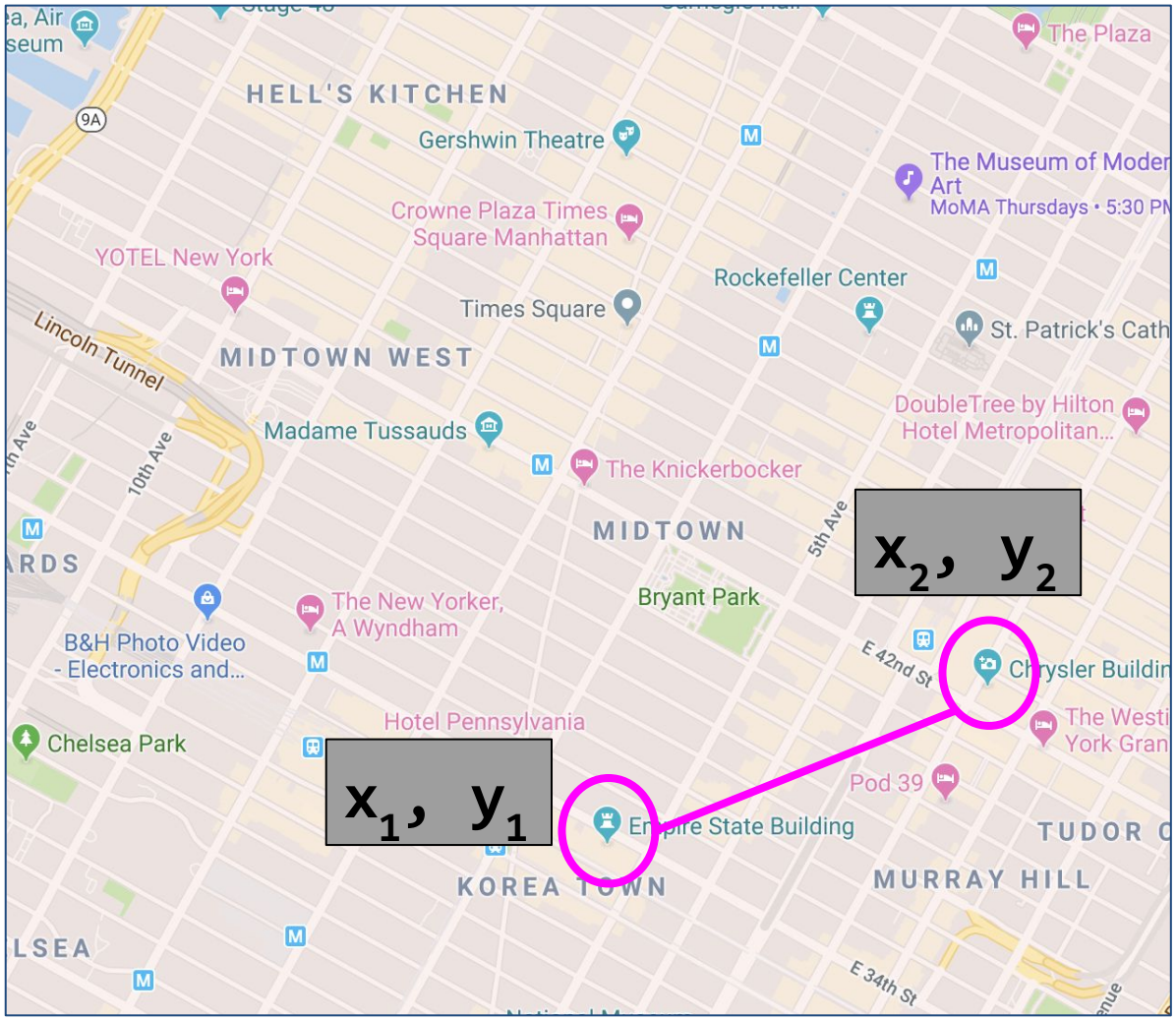
Did that print 7?

2. Euclidean Distance

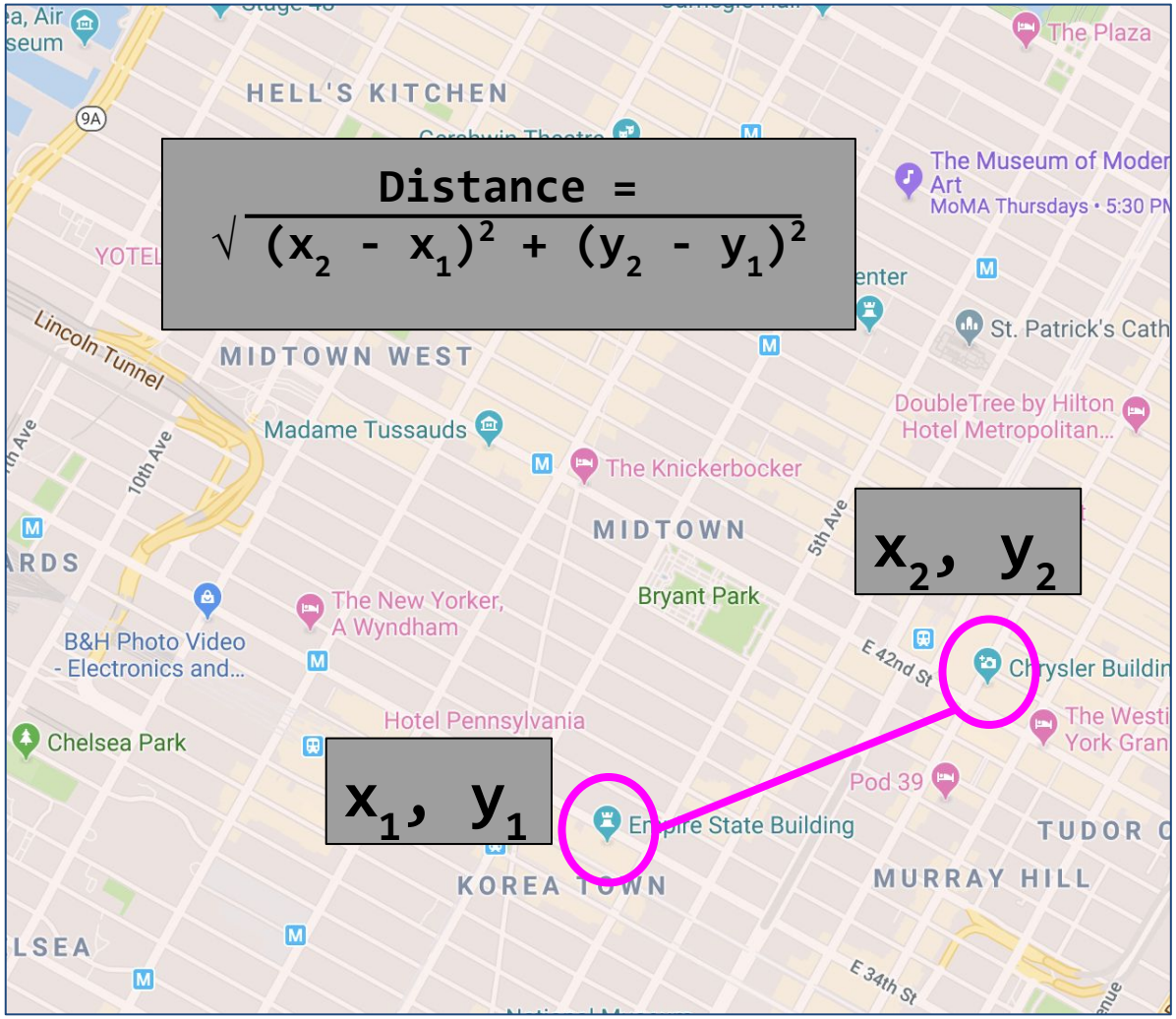
2. Euclidean Distance



2. Euclidean Distance



2. Euclidean Distance



2. Euclidean Distance

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2. Euclidean Distance

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
def euclidean(x1, y1, x2, y2):
    x_difference = (x2 - x1) ** 2
    y_difference = (y2 - y1) ** 2
    return (x_difference + y_difference) ** 0.5
```

2. Euclidean Distance

How did we do?

```
def euclidean(x1, y1, x2, y2):
    x_difference = (x2 - x1) ** 2
    y_difference = (y2 - y1) ** 2
    return (x_difference + y_difference) ** 0.5

distance = euclidean(0, 0, 3, 4)
print(distance)
```

2. Euclidean Distance

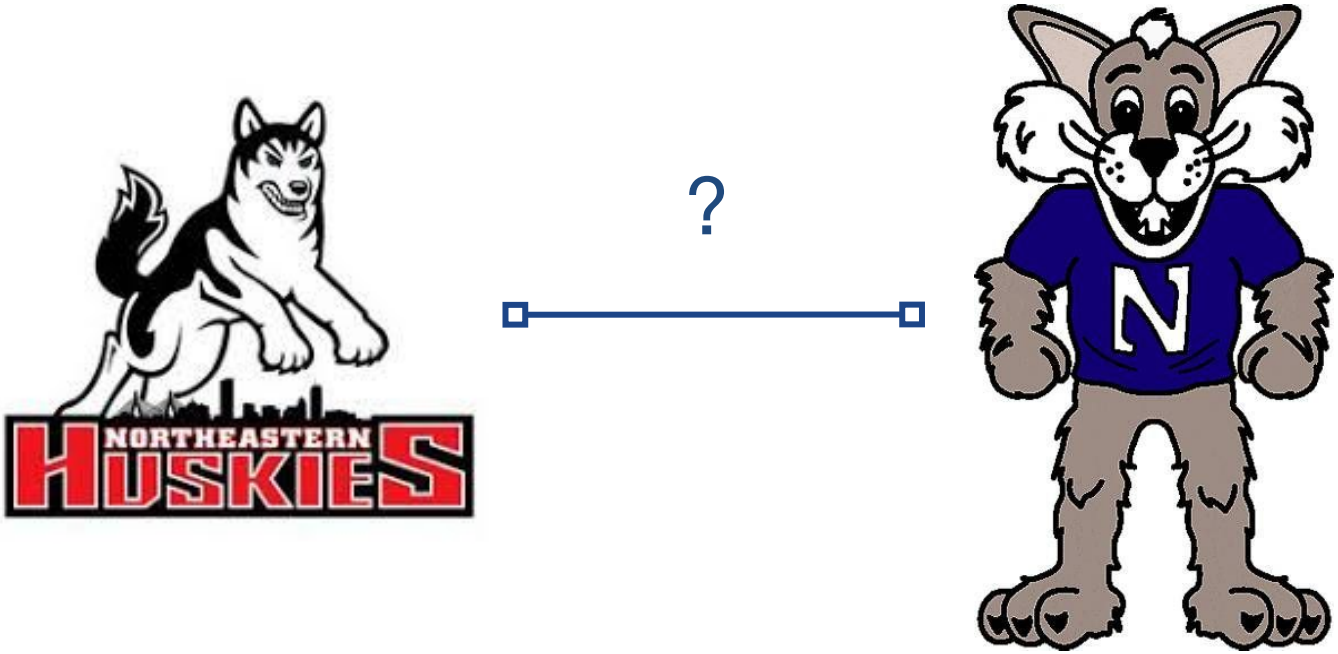
How did we do?

```
def euclidean(x1, y1, x2, y2):
    x_difference = (x2 - x1) ** 2
    y_difference = (y2 - y1) ** 2
    return (x_difference + y_difference) ** 0.5

distance = euclidean(0, 0, 3, 4)
print(distance)
```

Did that print 5.0?

3. Hamming Distance



3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

3. Hamming Distance

N	O	R	T	H	E	A	S	T	E	R	N

N	O	R	T	H	W	E	S	T	E	R	N

Hamming Distance = 2

3. Hamming Distance

```
Distance =
number of letters where
the two words differ
```

3. Hamming Distance

Distance =
number of letters where
the two words differ

```
def hamming(word1, word2):
    sum = 0
    length = min(len(word1), len(word2))
    for i in range(length):
        if word1[i] != word2[i]:
            sum += 1
    return sum
```

3. Hamming Distance

How did we do?

```
def hamming(word1, word2):
    sum = 0
    length = min(len(word1), len(word2))
    for i in range(length):
        if word1[i] != word2[i]:
            sum += 1
    return sum

distance = hamming('northeastern', 'northwestern')
print(distance)
```

3. Hamming Distance

How did we do?

```
def hamming(word1, word2):
    sum = 0
    length = min(len(word1), len(word2))
    for i in range(length):
        if word1[i] != word2[i]:
            sum += 1
    return sum

distance = hamming('northeastern', 'northwestern')
print(distance)
```

Did that print 2?

Putting it Together

Finding the Nearest Neighbor

1. Choose your distance function
   - Euclidean
   - Manhattan
   - Hamming

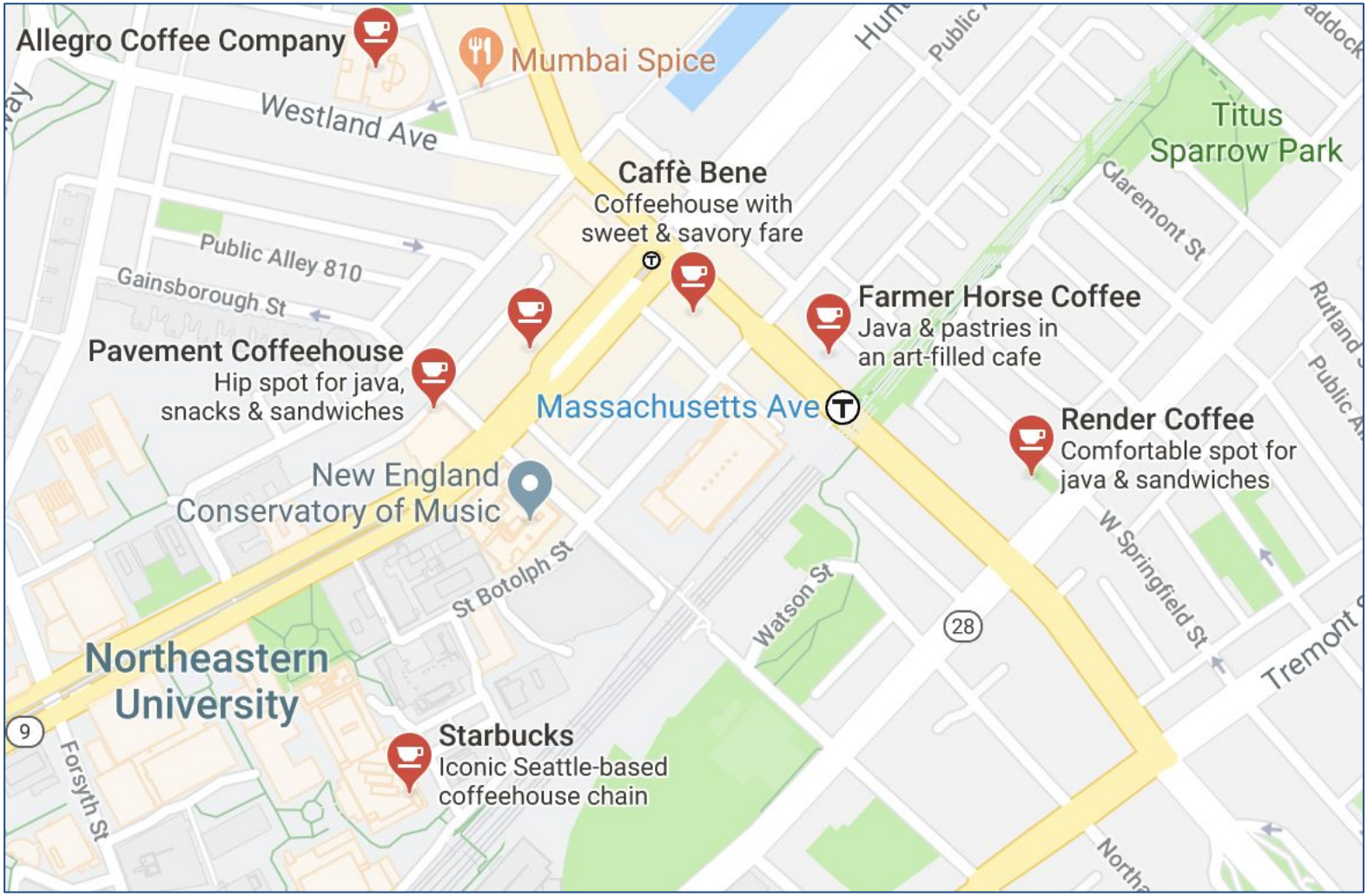
Finding the Nearest Neighbor

1. Choose your distance function
   - Euclidean
   - Manhattan
   - Hamming
2. Start with your known data
   - We know their *x*, *y* coordinates

Finding the Nearest Neighbor

1. Choose your distance function
   - Euclidean
   - Manhattan
   - Hamming
2. Start with your known data
   - We know their *x*, *y* coordinates
3. New data!
   - The others vote
   - Who am I closest to?

Data: Coffee Shops



Someone suggests a new coffee shop

Someone suggests a new coffee shop

Which one is it closest to?

Putting it Together

a. Start with existing x, y coordinates
b. Prompt the user for a new name, latitude, and longitude
c. Find the closest based on distance function
d. Go there and have coffee :)

Putting it Together

`bit.ly/wcoccoffee`

(Original data!)

Putting it Together

```
coffee_names = ["Pavement", "Caffe Bene", "Farmer Horse",
                "Starbucks", "Dunkin", "Peets",
                "Mystic", "Third Rail", "Home"]

coffee_places = [ [42.35, -71.18], [42.33, -73.09], [42.57, -71.88],
                  [43.33, -71.29], [44.34, -72.07], [41.88, -70.01],
                  [41.41, -71.11], [40.72, -73.99], [40.72, -73.98]]
```

Putting it Together

```
coffee_names = ["Pavement", "Caffe Bene", "Farmer Horse",
                "Starbucks", "Dunkin", "Peets",
                "Mystic", "Third Rail", "Home"]

coffee_places = [ [42.35, -71.18], [42.33, -73.09], [42.57, -71.88],
                  [43.33, -71.29], [44.34, -72.07], [41.88, -70.01],
                  [41.41, -71.11], [40.72, -73.99], [40.72, -73.98]]
```

Putting it Together

```
coffee_names = ["Pavement", "Caffe Bene", "Farmer Horse",
                "Starbucks", "Dunkin", "Peets",
                "Mystic", "Third Rail", "Home"]

coffee_places = [ [42.35, -71.18], [42.33, -73.09], [42.57, -71.88],
                  [43.33, -71.29], [44.34, -72.07], [41.88, -70.01],
                  [41.41, -71.11], [40.72, -73.99], [40.72, -73.98]]
```

Putting it Together

```
coffee_names = ["Pavement", "Caffe Bene", "Farmer Horse",
                "Starbucks", "Dunkin", "Peets",
                "Mystic", "Third Rail", "Home"]

coffee_places = [ [42.35, -71.18], [42.33, -73.09], [42.57, -71.88],
                  [43.33, -71.29], [44.34, -72.07], [41.88, -70.01],
                  [41.41, -71.11], [40.72, -73.99], [40.72, -73.98]]
```

Putting it Together

```
coffee_names = ["Pavement", "Caffe Bene", "Farmer Horse",
                "Starbucks", "Dunkin", "Peets",
                "Mystic", "Third Rail", "Home"]

coffee_places = [ [42.35, -71.18], [42.33, -73.09], [42.57, -71.88],
                  [43.33, -71.29], [44.34, -72.07], [41.88, -70.01],
                  [41.41, -71.11], [40.72, -73.99], [40.72, -73.98]]
```

Add this to your code

Putting it Together

Look up a new coffee shop! We'll need:

- Name
- Latitude
- Longitude

Putting it Together

```
new_cafe = input("What's the new cafe?")
lat = float(input("Latitude?"))
long = float(input("Longitude?"))
```

(New one!)

Putting it Together

```
closest_index = 0
closest_distance = manhattan(coffee_places[0][0],
                             coffee_places[0][1],
                             lat, long)
for i in range(len(coffee_places)):
   dist = manhattan(coffee_places[i][0], coffee_places[i][1],
                    lat, long)
   if dist < closest_distance:
      closest_index = i
      closest_distance = dist
```

(Find the closest -- manhattan)

Putting it Together

```
closest_index = 0
closest_distance = manhattan(coffee_places[0][0],
                             coffee_places[0][1],
                             lat, long)
for i in range(len(coffee_places)):
   dist = manhattan(coffee_places[i][0], coffee_places[i][1],
                    lat, long)
   if dist < closest_distance:
      closest_index = i
      closest_distance = dist
```

Assume the first one is closest

(Find the closest -- manhattan)

Putting it Together

```
closest_index = 0
closest_distance = manhattan(coffee_places[0][0],
                             coffee_places[0][1],
                             lat, long)
for i in range(len(coffee_places)):
    dist = manhattan(coffee_places[i][0], coffee_places[i][1],
                     lat, long)
    if dist < closest_distance:
        closest_index = i
        closest_distance = dist
```

Look through all of the original data

(Find the closest -- manhattan)

Putting it Together

```
closest_index = 0
closest_distance = manhattan(coffee_places[0][0],
                             coffee_places[0][1],
                             lat, long)
for i in range(len(coffee_places)):
    dist = manhattan(coffee_places[i][0], coffee_places[i][1],
                     lat, long)
    if dist < closest_distance:
        closest_index = i
        closest_distance = dist
```

Get the manhattan distance from the new place

(Find the closest -- manhattan)

Putting it Together

```
closest_index = 0
closest_distance = manhattan(coffee_places[0][0],
                             coffee_places[0][1],
                             lat, long)
for i in range(len(coffee_places)):
   dist = manhattan(coffee_places[i][0], coffee_places[i][1],
                    lat, long)
   if dist < closest_distance:
      closest_index = i
      closest_distance = dist
```

Is it closer?
Save it!

(Find the closest -- manhattan)

Putting it Together

```
print("The closest coffee house by manhattan is...")
print(coffee_names[closest_index])
```

(Announce the answer!)

Keep Going!

- It works for Manhattan? Try Euclidean!

Keep Going!

- It works for Manhattan? Try Euclidean!
- It works for Euclidean? Try Hamming!
  - It's a little different... use names instead of latitude/longitude

Keep Going!

- It works for Manhattan? Try Euclidean!
- It works for Euclidean? Try Hamming!
  - It's a little different... use names instead of latitude/longitude
- Find the 2 nearest neighbors instead of just one

The End :)

Questions?