CIS 1100

Information Representation
& Data Visualization
(Lecture One)

Python
Fall 2024
University of Pennsylvania

Updates and Reminders

- Apply to be a TA by 11:59pm tonight
  - link is on Ed
  - no late days accepted :)
- HW9 Released on Course Website
  - Due Dec 9, but start early
  - No late days, no drops
  - (technically only the first part is on the website, but that's plenty and the other part will follow)
- Midterm 2 grades out early next week

Questions?

Information Representation

Basically, symbolism! What can it mean when I use X to represent something?

We'll talk about this in terms of:

- data types
- graphics & graphical markers (visual symbols)

Representation: Types

An `int` is a data type for integral (whole) numbers.

The typical interpretation of an `int` is a **quantity**: I have `10` eggs in my refrigerator, or there are `103` students in this class.

Divide **(C12)** in half vertically. On the left, write as many things as you can think of that an `int` can be used to represent. (Feel free to brainstorm with a partner.)

What Did You Come Up With?

`2158983500` (which can have a few meanings...)

`1100`

`-1`

Representation: Types

A `str` is a data type for sequences of characters.

On the right side of **(C12)**, list at least eight things that a `str` can represent.

Representation: Types

From examples that we've done in class:

- Names for people
- Titles (of songs, books, movies)
- Genres (of songs, books, movies)
- Types of cuisine (of restaurants)
- Line names for transit routes (e.g. "M4" bus)
- Histories about injuries/illnesses
- Place names
- **LISTS** of these things (lists of genres for a song, lists of transit routes serving a school)

Takeaway:

Programming is hard, not least because it's hard to keep straight what different variables & types are trying to ***be!***

In Caesar:

- a "message" was both a list of ints and a string
- a list of ints could be both a "cipher" and a "message"

➡️ different meanings can be encoded with different types, and the same type can encode different meanings.

I will tell you a terrible secret: language is punishment. Language must encompass all things and in it all things must again transpire according to guilt and the degree of guilt. — *Malina* by Ingebord Bachmann

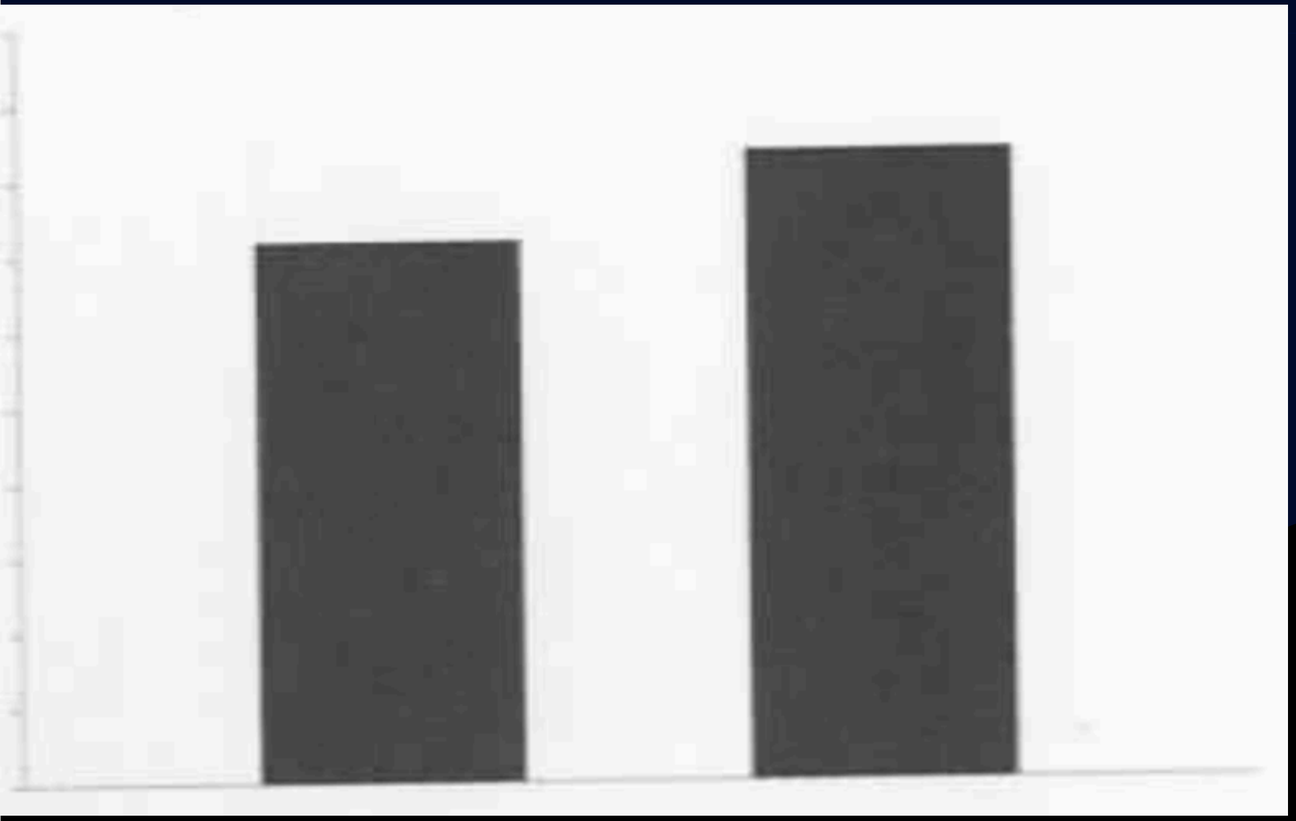
And meaning, after all, is a kind of luck - - some things just shine with it, and no one knows why. — *Priestdaddy* by Patricia Lockwood

Representation in *Visual* Language

It's hard to be clear in programming languages. It's also hard to be clear in natural languages.

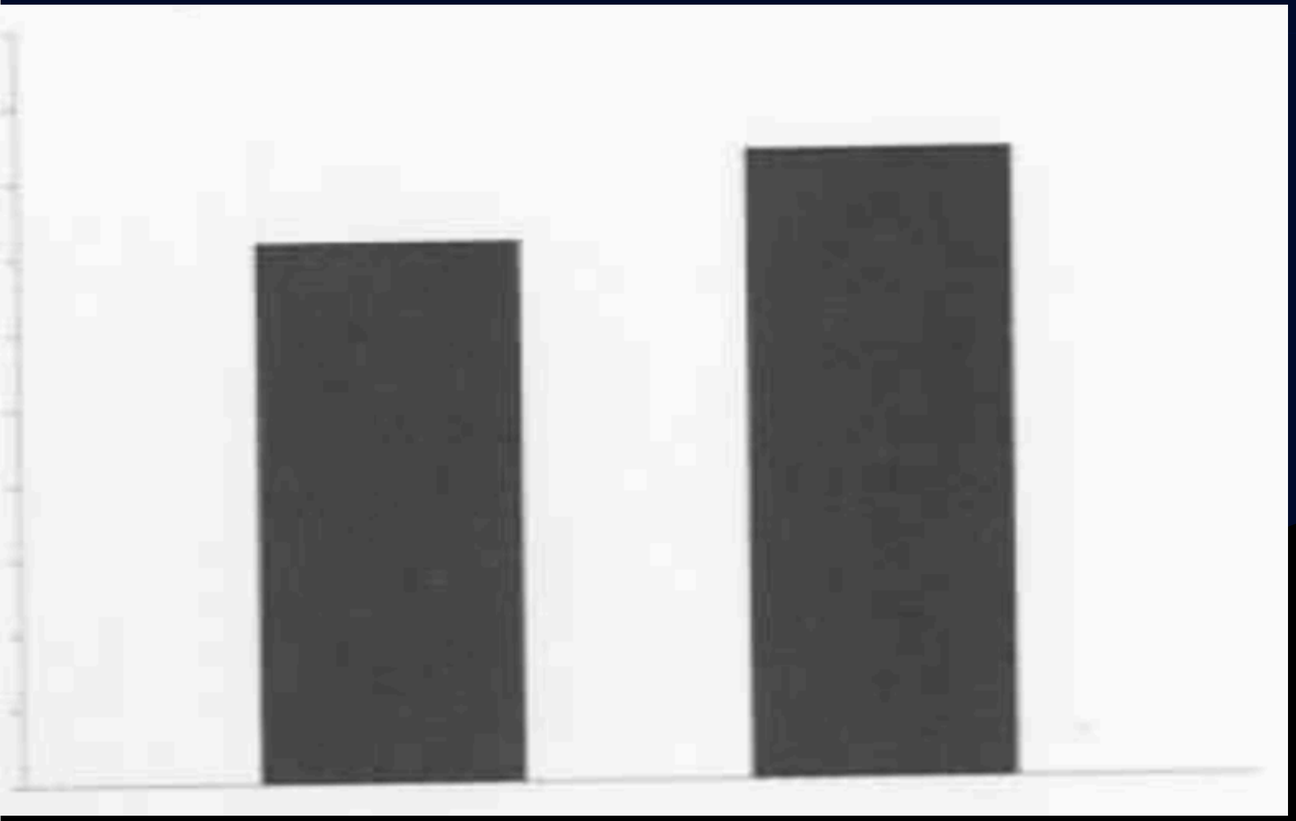
Let's talk about how it's **ALSO** hard to be clear when drawing pictures...

Exercise (L11)



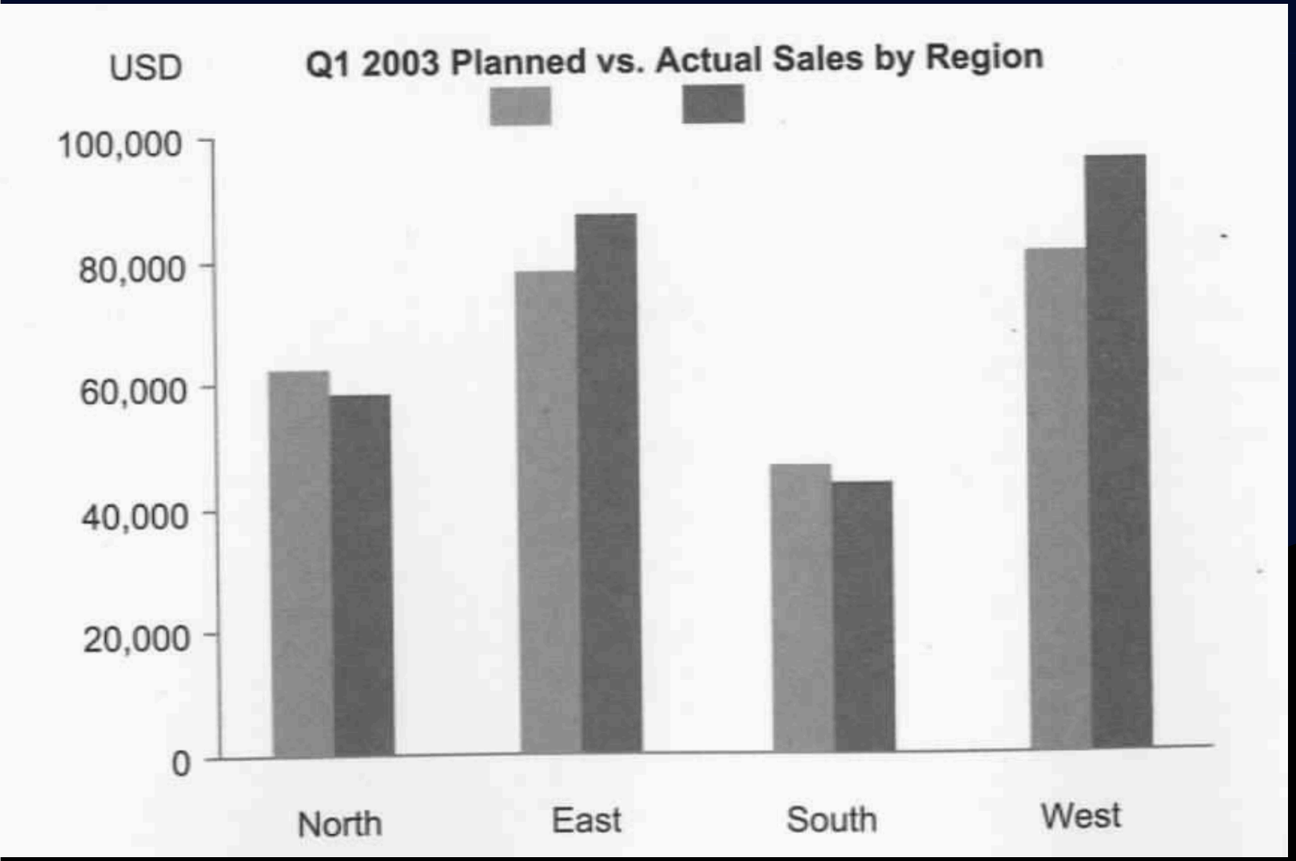
Briefly: *When you look at the two dark rectangles below, what do you notice and what meanings come to mind?*

Bars



- heights, and differences between them
- weight (width) and contrast
  from the background
- position:
  - along the x-axis, separation
  - along the y-axis,
    alignment at the bottom

Exercise



In one or two words...

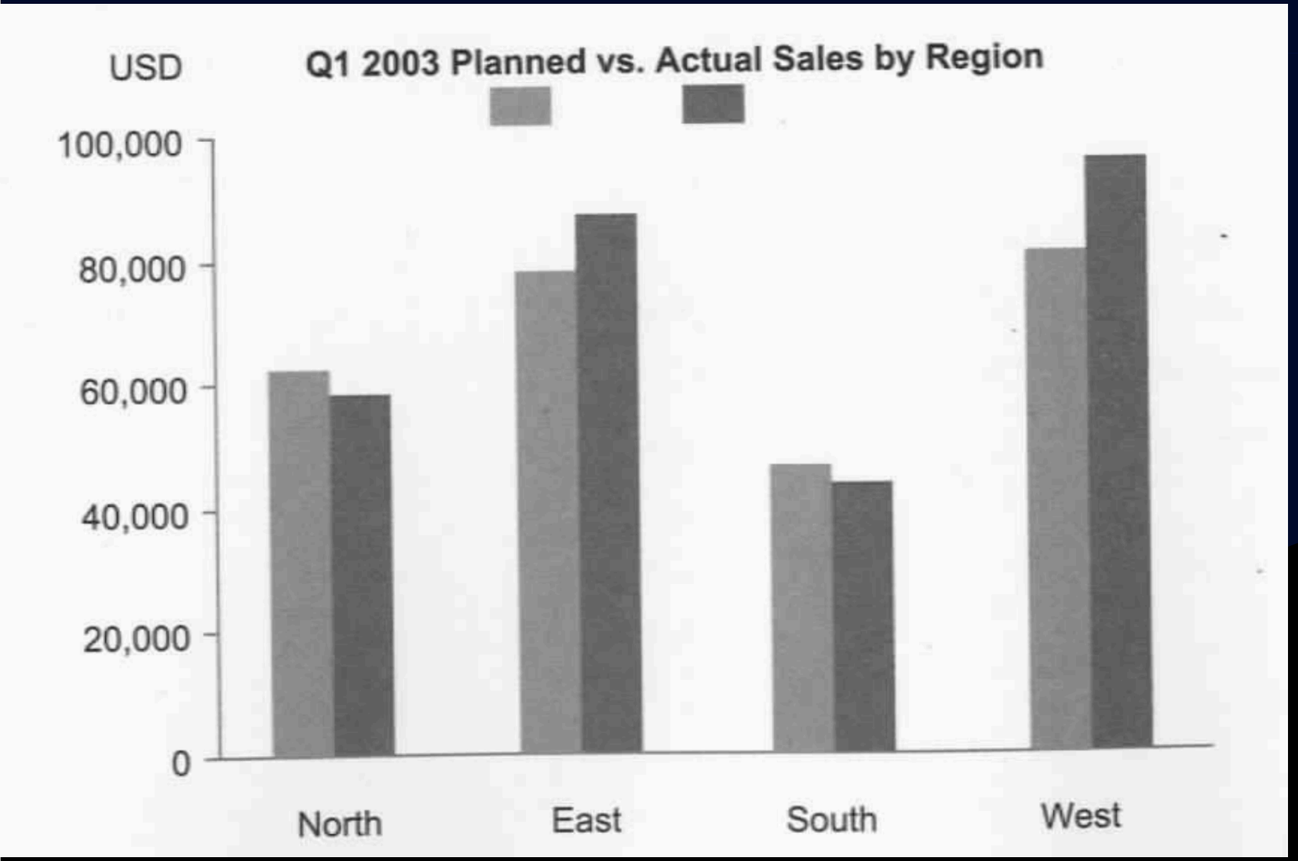
(S7) What does the height of a bar encode (represent)?

(S8) What does the width of a bar encode?

(S9) What does the x-position of a bar encode?

(S10) What does the color of a bar encode?

Exercise




```
pd.filled_rectangle(x, y, hw, hh)
```

Mark all that apply:

A: `x`, B: `y`, C: `hw`, D: `hh`: E: other

(M1) Which parameters are used to encode the height of a bar?

(M2) Which are used to encode the width?

(S9) Which are used to encode the x-pos?

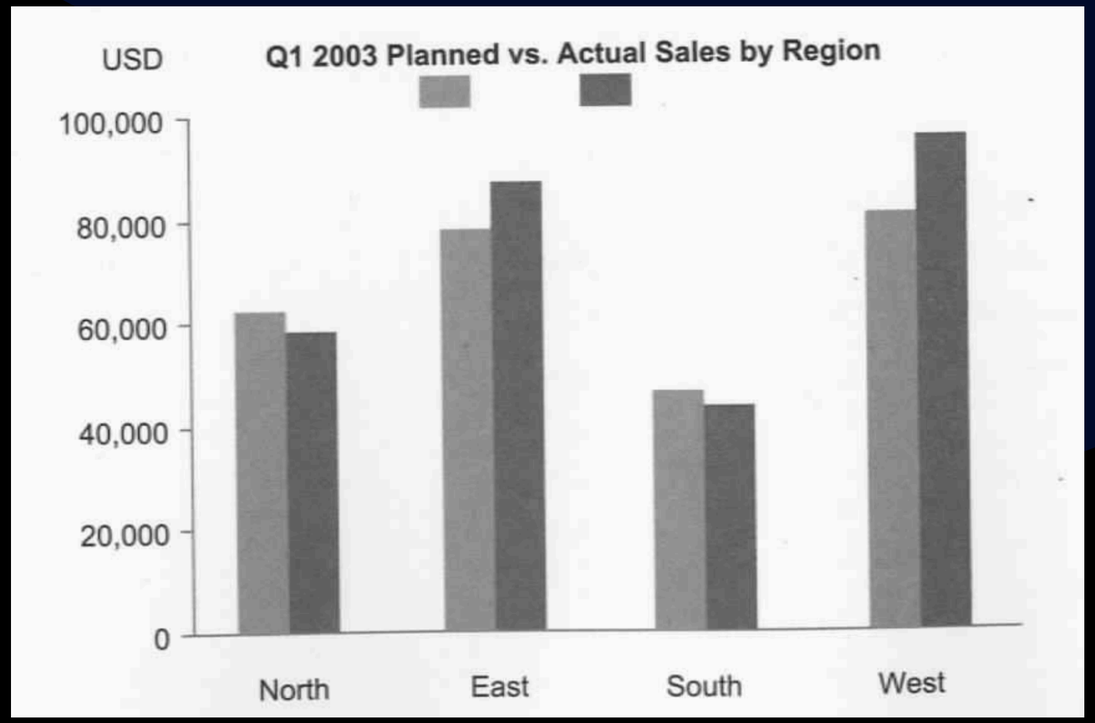
(S10) Which are used to encode the height?

Can We Replicate a Bar Chart Together?

Dictionary of exam 1 average scores and final exam average scores:

```
scores_dict = {
    "23fa": [80.97, 80.73],
    "24sp": [76.73, 68.52],
    "22fa": [71.98, 75.79],
    "23sp": [78.61, 65.53],
}
```

```
def paired_bar_chart(scores, x_min=0.1, x_max=0.9, y_min=0.1, y_max=0.9):
    y_range = y_max - y_min
    x_range = x_max - x_min
    ...
```



`y_min` and `y_max` correspond to the y-coordinates for the y axis line and the maximum allowable height for a bar (at $100k).

```
scores_dict = {
    "23fa": [80.97, 80.73],
    "24sp": [76.73, 68.52],
    ...
}
```

(L13) Can you write an expression to calculate the *half-height* of a bar in this chart?

CIS 1100

Information Representation
& Data Visualization
(Lecture Two)

Python
Fall 2024
University of Pennsylvania

Recap From Check-In

What did you think of the Bertin reading for the check-in?

Data Types by Meaning

1. **Nominal/Qualitative:** categories that are similar but not universally orderable
    i. e.g. Strings as names (of dishes on a menu, of song titles)
2. **Ordered:** categories whose elements can be ordered in a single way
    i. these elements are in fact defined by their *differences* rather than similarities
    ii. e.g. Strings as names of *students* on an attendance list or `datetime` objects semesters
3. **Quantitative:** categories with elements that can be compared in arithmetic terms (relative increases)
    i. **Interval** categories are those with no absolute zero, e.g. times of day, human temperature readings
    ii. **Ratio** categories are thsoe with an absolute zero, e.g. prices, elevation

Category Practice

A: Nominal, B: Ordered, C: Q Interval, D: Q Ratio

(M1) Movie review scores out of five

(M2) Movie genres

(M3) Movie titles

(M4) User IDs as modeled by strings of integers, e.g. 314 or 43242

(M5) Movie box-office sales

Why Talk About These Categories?

They give heuristics for the kinds of **marks** that you would draw in a graphic and the ways that you modulate these to convey information.

Marks & Channels

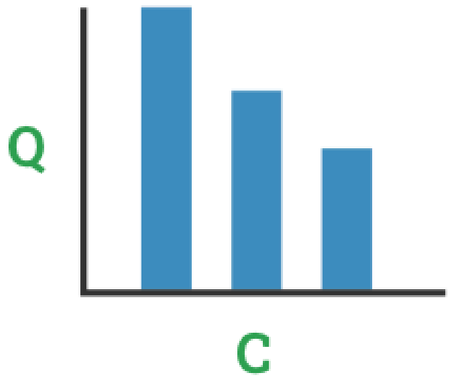
Marks are the geometric shapes that make up a graphic

- the stuff you draw with PennDraw commands like `rectangle` or `line` or `point`

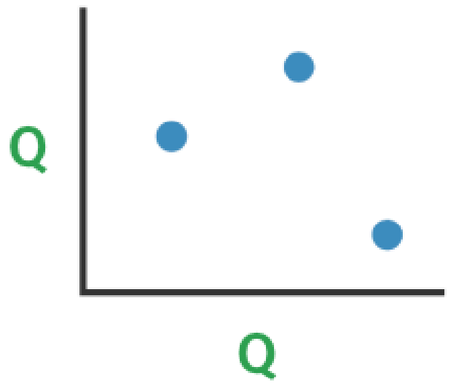
Channels are the ways that we modify the marks, including:

- positions, size, area, or tilt/angle (i.e. parameters of the `pd` calls themselves)
- color and thickness, which are changed by separate calls to `set_pen_color` and `set_pen_radius`

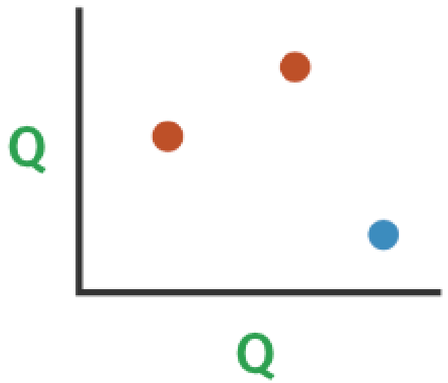
A Blatantly Plagiarized Example Courtesy of COMS 4995 at Columbia



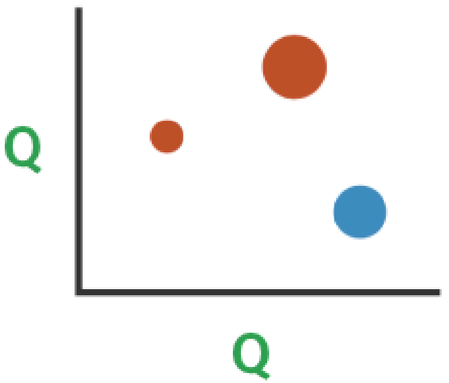
Mark: **line**



Mark: **point**



Mark: **point**
Channels:
Color: **C**



Mark: **point**
Channels:
Color: **C**
Size: **Q**

Recreating These in Python with PennDraw

We need to convert *values* into *marks* with different *channels!*

```
temperature_readings = [
  {"hour" : 1, "temperature" : 100.4, "symptomatic" : True},
  {"hour" : 3, "temperature" : 101.1, "symptomatic" : True},
  {"hour" : 4, "temperature" : 99.1, "symptomatic" : False},
]
```

This can be helped using **scales** of different kinds, which are functions that map input values of different domains into output values that can be made sense of in our drawing programins.

Linear Scales

Given an input value, the range that the input comes from, and the range of values that can be output, return the input mapped to the output range.

```
def scale_linear(value, min_input, max_input, min_output, max_output):
  fraction_input = (value - min_input) / (max_input - min_input)
  output_range = max_output - min_output
  return fraction_input * output_range + min_output
```

e.g. If measurements are always taken between `0` and `6` hours after illness offset, then a reading at hour `3` is precisely in the middle of the input range. If the output range is supposed to represent coordinates on a screen between `0.1` and `0.9`, then the matching output point is halfway between them at `0.5`.

Practice with Linear Scaling

(S7) We have a measurement `18` from an input range of `[0, 24]` and we want to calculate the corresponding output in the range `[0, 100]`. What is the output?

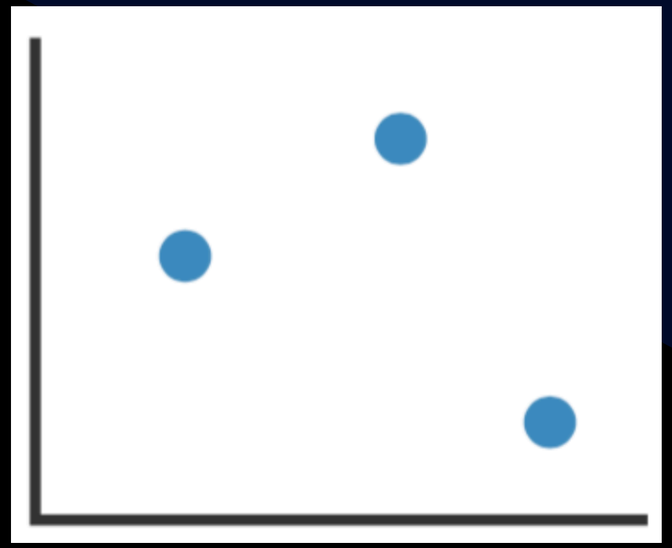
```
scale_linear(18, 0, 24, 0, 100)
```

(S8) What value of `input_val` would cause this function call to return `4`?

```
scale_linear(input_val, 1, 3, 3, 5)
```

Back To Work

We want to plot *hours since onset* as the x-coordinate.



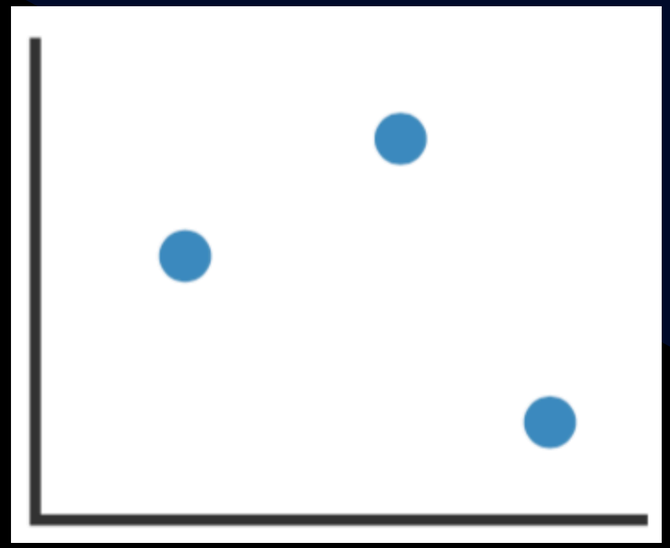
```
readings = [
  {"hour" : 1, "temperature" : 100.4,
   "symptomatic" : True},
  {"hour" : 3, "temperature" : 101.1,
   "symptomatic" : True},
  {"hour" : 4, "temperature" : 99.1,
   "symptomatic" : False},
]
```

(S9) To calclulate the x coord of a point from an individual `reading` dict, fill in the blank:

```
for reading in readings:
    x_pos = scale_linear(______, 0, 6, x_min, x_max)
    ...
```

Back To Work

We want to plot *temperature* as the y-coordinate.



```
readings = [
  {"hour" : 1, "temperature" : 100.4,
   "symptomatic" : True},
  {"hour" : 3, "temperature" : 101.1,
   "symptomatic" : True},
  {"hour" : 4, "temperature" : 99.1,
   "symptomatic" : False},
]
```

(L11) To calclulate the y coord of a point from an individual `reading` dict, finish the line:

```
for reading in readings:
    x_pos = scale_linear(reading["hour"], 0, 6, x_min, x_max)
    y_pos = scale_linear(_______, 98.6, 103, ____, ____)
    ...
```

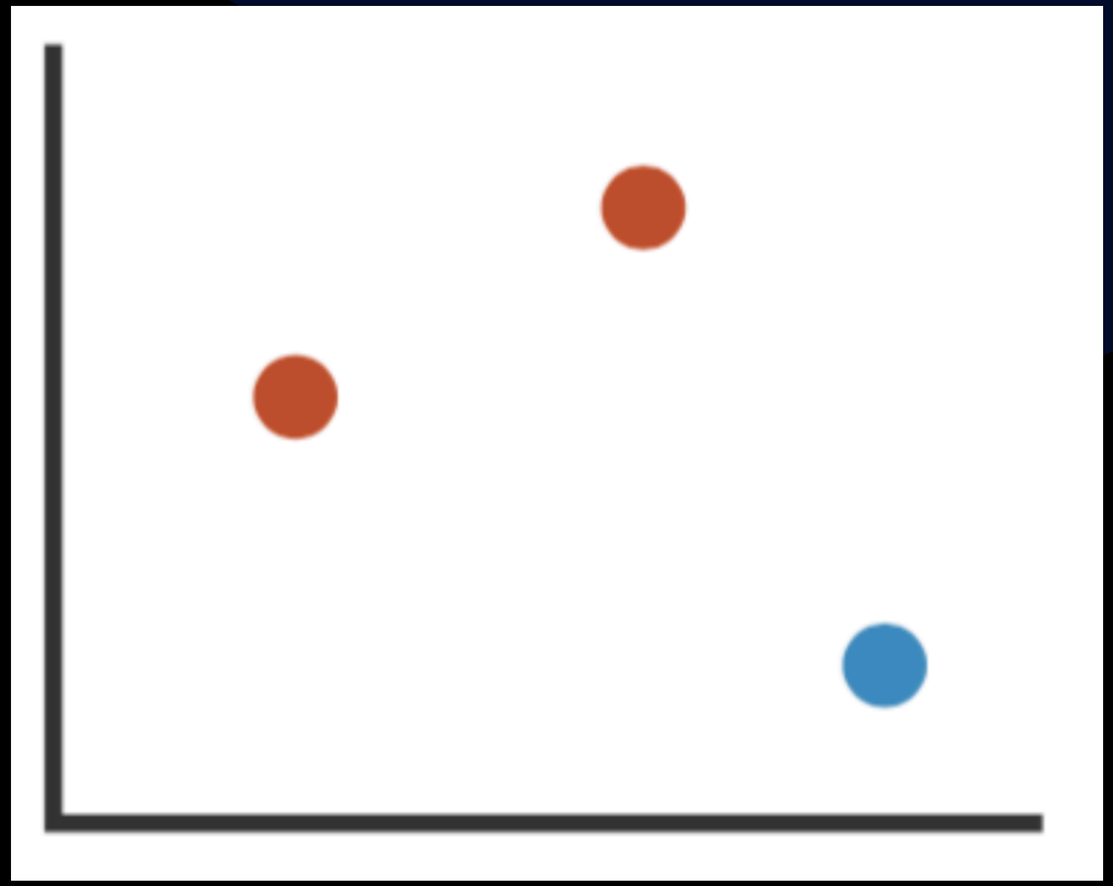
We Did It

```
import penndraw as pd

x_min, x_max, y_min, y_max = 0.1, 0.9, 0.1, 0.9

# axes
pd.line(x_min, y_min, x_min, y_max)
pd.line(x_min, y_min, x_max, y_min)

for reading in readings:
  x_pos = scale_linear(reading["hour"], 0, 6, x_min, x_max)
  y_pos = scale_linear(reading["temperature"], 98.6, 103, y_min, y_max)
  pd.filled_circle(x_pos, y_pos, 0.025)

pd.run()
```

Can We Do More Of "It"?

There's a channel that we didn't implement yet...

We want to use *symptomatic* as the color channel.



```
readings = [
  {"hour" : 1, "temperature" : 100.4,
   "symptomatic" : True},
  {"hour" : 3, "temperature" : 101.1,
   "symptomatic" : True},
  {"hour" : 4, "temperature" : 99.1,
   "symptomatic" : False},
]
```

Feeling Colorful

(C12)

```
for reading in readings:
  x_pos = scale_linear(reading["hour"], 0, 6, x_min, x_max)
  y_pos = scale_linear(reading["temperature"], 98.6, 103, y_min, y_max)

  """
  TODO: PUT SOMETHING HERE TO MAKE THE COLORS MATCH
  """

  pd.filled_circle(x_pos, y_pos, 0.025)
```

Why Did We Do All This Data Viz Stuff?

1. Discussing the meanings of symbols, types and programming constructs
2. Thinking critically about graphics & processing visually encoded information
3. Identifying relationships between visual representations
   and mathematical rules & formulae that define them
4. Connecting PennDraw and its systems to new applications (outside of games & animation)