Programming & Computational Thinking

Abstraction & Generalization

Computational Thinking

Remember talking about computational thinking and these key principles?

- abstraction
- generalization
- composition & decomposition
- creativity
- data and information
- algorithms

Review - Abstraction

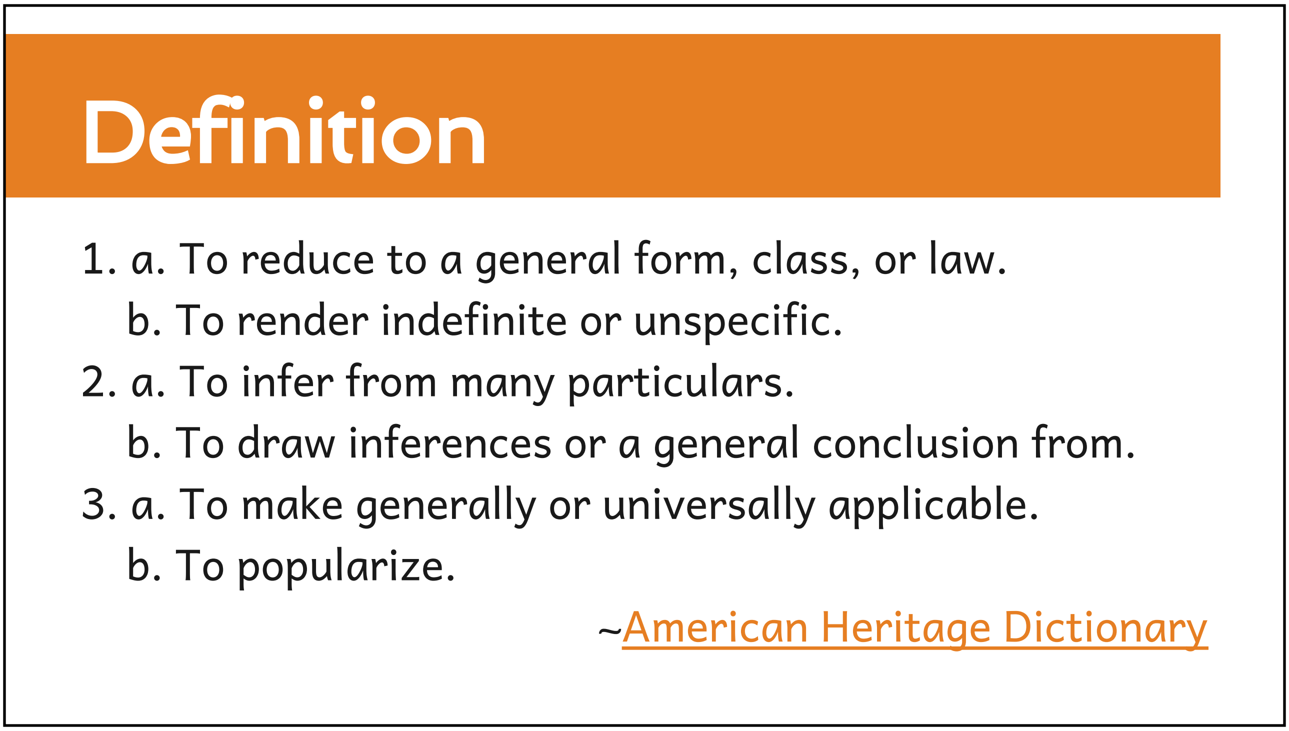
We discussed abstraction last week.

- What does it mean?
- How is it used in everyday examples?
- How about with computer or IT examples?

Generalization

This week we'll talk about generalization.

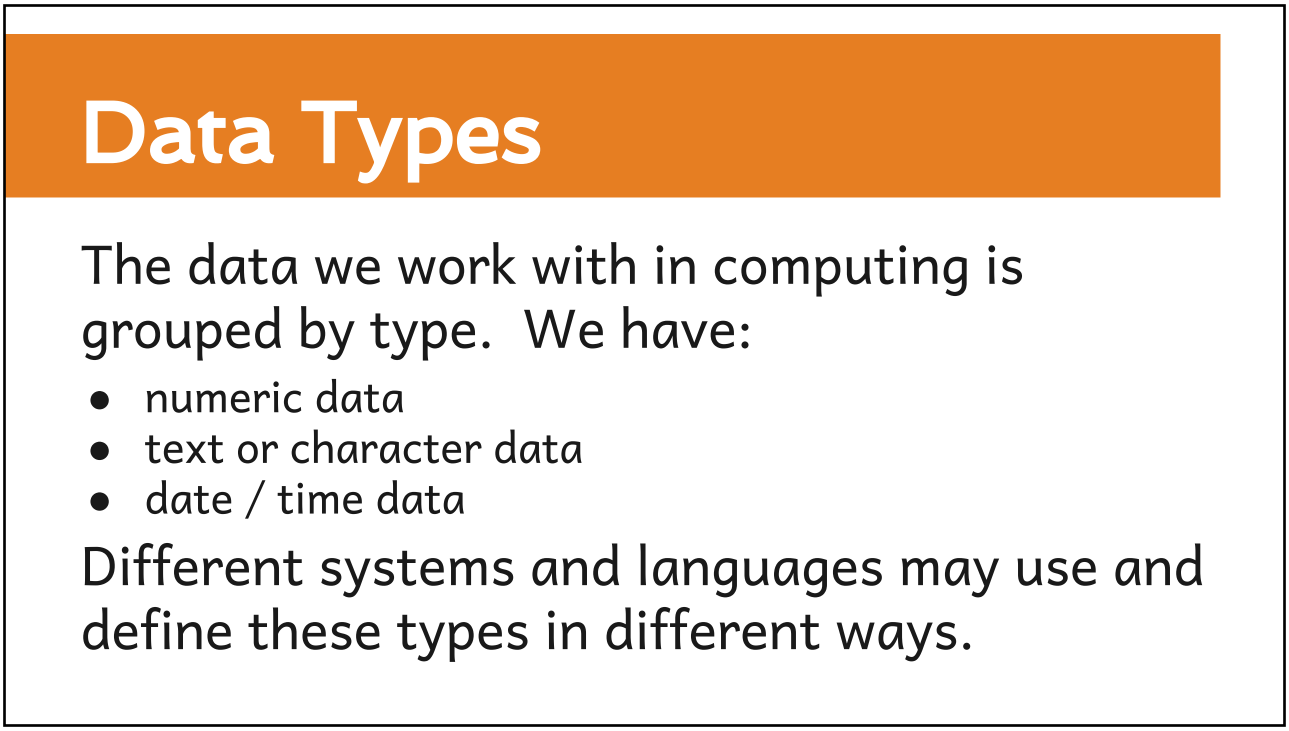
What does it mean to generalize?



Definition

1. a. To reduce to a general form, class, or law.
   b. To render indefinite or unspecific.
2. a. To infer from many particulars.
   b. To draw inferences or a general conclusion from.
3. a. To make generally or universally applicable.
   b. To popularize.

~American Heritage Dictionary



Data Types

The data we work with in computing is grouped by type. We have:

- numeric data
- text or character data
- date / time data

Different systems and languages may use and define these types in different ways.

Numeric Data Types

Different types of numbers:

- Integers
  - Examples: 1, 2, 4, 0, -10
- Floating point (have a decimal point)
  - Examples: 1.2, 4.5, -3.0

Different ways to represent them:

binary, hex, decimal

Character Data Types

Ways to represent a single character:

- EBCDIC, ASCII, Unicode
- All go back to a numeric value

- Words and sentences are just *strings* or sequences of individual characters.
- Number 1 is different from a character 1.

Objects

Represent more complex data and systems

For example a generic Course has a:

- Course name
- Department
- Course number
- Section number
- Number of credits
- Description

Generalizing Actions

Data types & objects are things or *nouns*

We can also generalize behaviors.

- morning routine
- registering a student for a class
- making a sandwich

Sandwiches for lunch

My school lunches includes a nice variety of sandwiches:

- ham (almost always)
- turkey (often)
- chicken (sometimes)
- meatloaf (increasingly rare)

Sandwich recipes

Let's look at my mom's sandwich recipes:

1. Ham Sandwich: Take a slice of bread, add 2 slices of ham, top it with another slice of bread.
2. Turkey Sandwich: Take a slice of bread, add 2 slices of turkey, top it with another slice of bread.
3. Chicken Sandwich: Take a slice of bread, add some bits of chicken, top it with another slice of bread.
4. Meatloaf Sandwich: Take a slice of bread, add 2 slices of meatloaf, top it with another slice of bread.

Are these really 4 different recipes?

Sandwich recipes

Let's look at my mom's sandwich recipes:

1. Ham Sandwich: Take a slice of bread, add **2 slices of ham**, top it with another slice of bread.
2. Turkey Sandwich: Take a slice of bread, add **2 slices of turkey**, top it with another slice of bread.
3. Chicken Sandwich: Take a slice of bread, add **some bits of chicken**, top it with another slice of bread.
4. Meatloaf Sandwich: Take a slice of bread, add **2 slices of meatloaf**, top it with another slice of bread.

They look very similar, so let’s generalize.

General Sandwich recipe

The only difference is the meat:

1. Take a slice of bread
2. Add some meat
3. Top it with another slice of bread.

What benefits do you see in making the recipe generic?

Functions

A function is a way to generalize an action

Examples might be:

- making a sandwich (I'm sure somewhere robots do this...)
- adding two numbers
- saying hello to someone in the chat room (Slackbot)
- drawing a shape

Good functions

A good function has a few characteristics:

- descriptively named
- it is generalized - repeatable & reusable
- conceptually, it does a single thing
- it is independent - testable in isolation

Composition

When you have a function that does one thing, you might want to use it as part of another function that does something bigger.

Using one function inside of another function is composition.

Composition Example

- The “make sandwich” function might be used by the “make lunch” function
- The “make lunch” function might also be used by the “get ready for school” function
- It’s even possible the “get ready for school” function does nothing but use other functions.

Lab Intro - Shapes

In lab today we'll be looking at ways to generalize as we draw shapes.

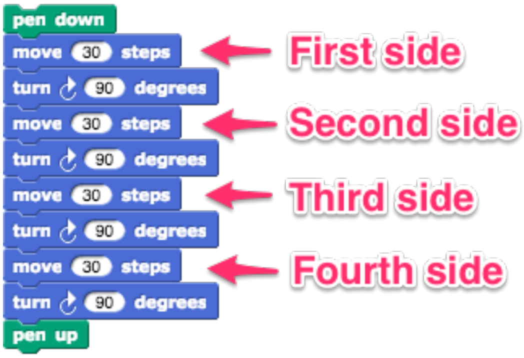
We'll look at:

- ways to repeat commands
- ways to make them more general & multi-purpose

Draw a square

Let's look at some *Snap!* code to draw a square.

These block should look familiar from last week.

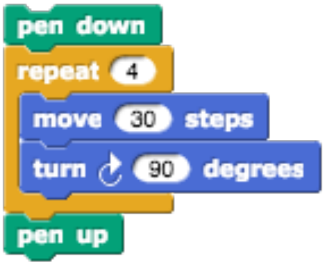


```
pen down
move (30) steps
turn ↻ (90) degrees
move (30) steps
turn ↻ (90) degrees
move (30) steps
turn ↻ (90) degrees
move (30) steps
turn ↻ (90) degrees
pen up
```

Generalize with Repeat

We do the same thing to draw each of the 4 sides of the square.

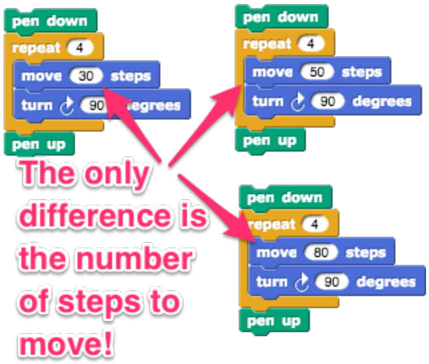
So we can generalize by repeating the steps for drawing one side 4 times.



```
pen down
repeat 4
    move 30 steps
    turn ↻ 90 degrees
pen up
```

Making it bigger…

What if we want to draw bigger squares?



```
pen down
repeat 4
    move 30 steps
    turn ↻ 90 degrees
pen up
```

```
pen down
repeat 4
    move 50 steps
    turn ↻ 90 degrees
pen up
```

```
pen down
repeat 4
    move 80 steps
    turn ↻ 90 degrees
pen up
```

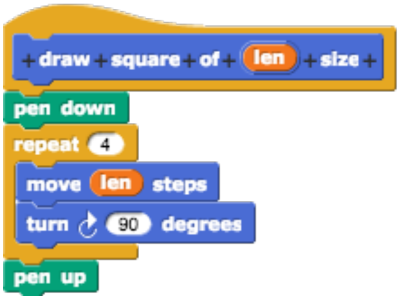
The only difference is the number of steps to move!

This is somewhat similar to our sandwich recipe.

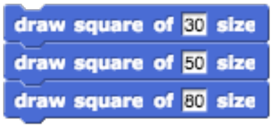
Instead of a different meat, we have a different number of steps to move.

Generalize with Functions

Here we have a "draw square" block that takes a *variable* number for the side length.



```
+draw+square+of+ (len) +size+
pen down
repeat (4)
  move (len) steps
  turn ↻ (90) degrees
pen up
```



```
draw square of [30] size
draw square of [50] size
draw square of [80] size
```

Functions in *Snap!*

- *Snap!* functions are called custom blocks.
- The orange *variable* piece is called an input value or parameter.
- You make a custom block by using the button on the Variables pallette.
- You'll learn how to do this in tonight's lab.

A general shape?

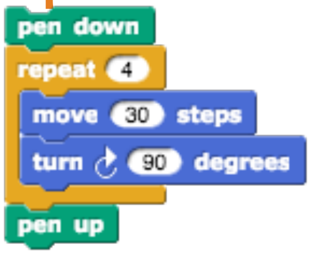
Looking at what we did to generalize drawing a square to allow us to make different sized squares, you might wonder how far can we take this?

Can we generalize a function to draw any shape?

Making a triangle

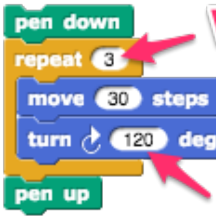
To generalize, look at specific examples.
Look for what is the same and what is different.

Square:



```
pen down
repeat (4)
    move (30) steps
    turn ↻ (90) degrees
pen up
```

Triangle:



```
pen down
repeat (3)
    move (30) steps
    turn ↻ (120) degrees
pen up
```

We only need 3 sides

The angle is larger

Draw polygon

In lab, you’ll continue this exercise to examine more examples and come up with a generalization for drawing a polygon.

It will be able to have:

- different lengths for each side
- and any number of sides

Lab Time!

Remember your lab partner from last week? Time to hook up again for lab 2.

Lab 3 is homework - no class next week.

We’ll trade partners for lab 4 (next class).