Welcome to Lecture 4: Lists and Loops

1) Use Iclicker for attendance
2) Lecture Guide: tinyurl.com/S24CS10L4
3) Start at 10:10 (Berkeley Time)...but thank you for coming at 10!

Agenda / Topics

- L3 Review
- Intro to Lists
- Lists and Loops
- List Mutation

Lecture Guide: tinyurl.com/S24CS10L4

Announcements

- If you miss lecture, do the lecture quiz for attendance!
- Use Ed for questions
- Support Office Hours…
- Project 1 Due today

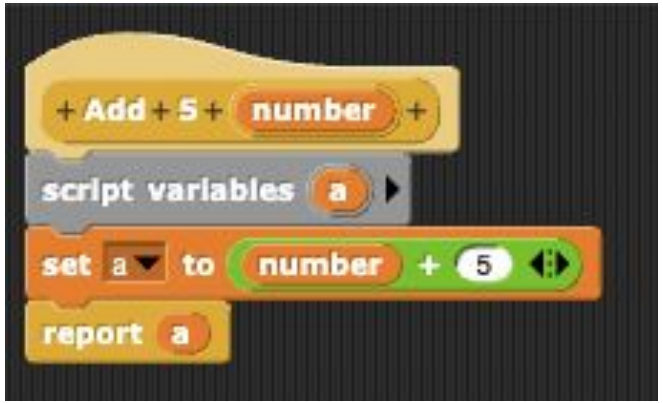
Lecture Guide: tinyurl.com/S24CS10L4

Review from Lecture 3 - Scope

- **Global Variables**
  - Global variables are declared outside of all functions.
  - **Scope**: They can be accessed and modified from any part of the program, including inside functions.
- **Local Variables (Script Variables)**
  - Local variables are declared within a function.
  - **Scope**: They can only be accessed within the function where they are defined.
  - Cannot be accessed directly outside the function.
    - However, their values can be returned by the function and used elsewhere.

Saving a Local Variable outside a Function

- Return the value of a local (script) variable outside the function
- Save the return value in a global variable



Iteration

- Iteration is a set of instructions that are executed repeatedly
- Here are some different iterative statements in Snap*!*

- All three of these iterative statements…
  - Will run everything inside of the loop and repeat it
  - Can be terminated early if a “report” block is called inside

Russian Nesting Dolls

Intro to Lists

- A Data Structure that holds individual data types
  - Text, Numbers, Booleans
  - Lists of Lists (2D lists)--> future lecture

- Ordered collections of values
  - Each item as a location/address in a list - called the “index”

Lecture Guide: tinyurl.com/S24CS10L4

Intro to Lists

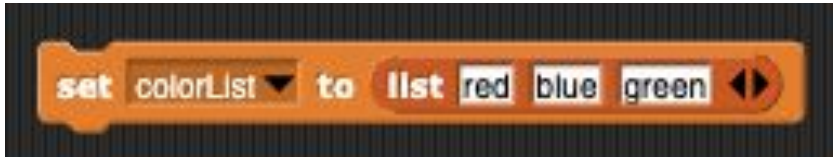
- Color list:
  `["red", "blue", "green"]`

Imagine a spreadsheet column….

	A
Index	ColorList
1	Red
2	Blue
3	Green

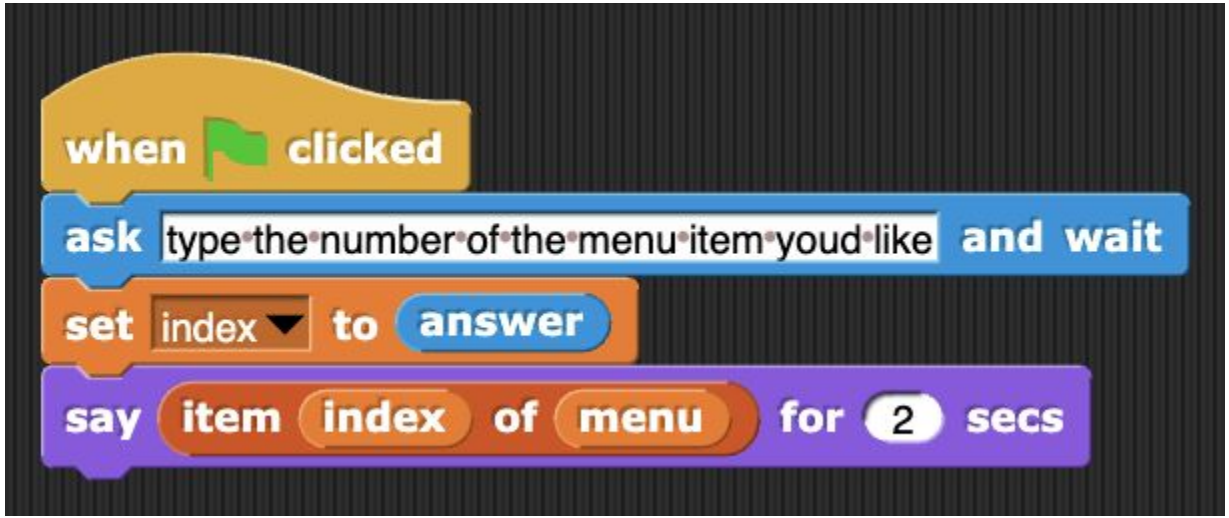
List Indexes

- Like an address -> points to the memory cell the element is stored in
- Access a specific item in a list



Dynamic List→ Get Index from User

*In this example we see that the index of the list can be a dynamic value that comes from the user.



LENGTH OF A LIST

- Length of a list = # items it contains
- Dynamically calculates length of list
    - If you add or remove an element, will adjust value automatically
- Built-in function - exists in most, if not all, modern programming languages

List with Random and Length of List

- Example: Magic 8 Ball

Lecture Guide: tinyurl.com/S24CS10L4

Questions?

Lists and Loops

- We can use loops to visit or manipulate every value in a list.
- A program can be written that uses a loop to deal with each thing in a list, no matter how many things are in the list.

Lecture Guide: tinyurl.com/S24CS10L4

List Iteration - For Each

- **Iteration** is the process of repeating certain instructions in a computer program.
- Lists are **iterable**: we can sequentially access (loop over) list items one-by-one.
- You dont need “length of list” here

List Iteration - For “i” Loops

- Iteration is the process of repeating certain instructions in a computer program.
- Lists are iterable: we can sequentially access (loop over) list items one-by-one.

WHAT WOULD SNAP! SAY?

```
set Colors to list Red Blue Green

for i = 1 to length of Colors
    say i for 2 secs
```

Lecture Guide: tinyurl.com/S24CS10L4

WHAT WOULD SNAP! SAY?

A. Red, Blue, Green

B. 1, 2, 3

- Notice: the "i" represents the index position within the color
- You need to add it inside the "item of block" to access the list

WHAT WOULD SNAP! SAY?

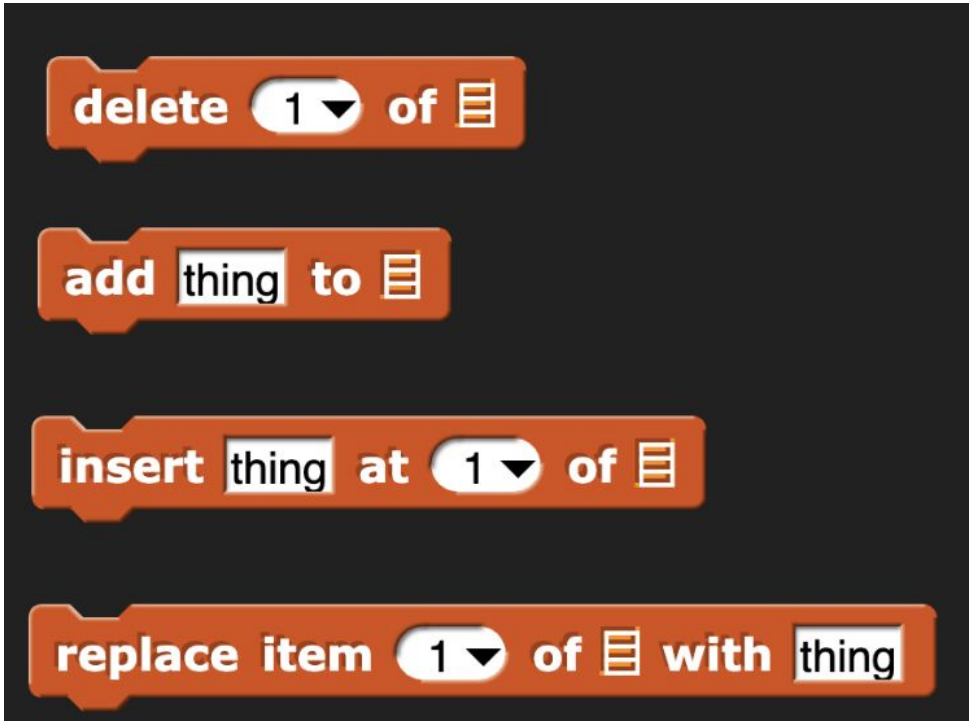
- Notice: the "i" represents the index position within the color
- You need to add it inside the "item of block" to access the list

Task: Multiply all Items in a list by 2

- Create a list of numbers
- Use a loop to return a new list with all the values multiplied by 2
- Hint:
  - Use a script variable to make and return a new list

List MUTABILITY

- In Snap!, there are four main blocks that allow us to mutate lists:



Add

*Default is to add the new entry to the end of the list

Task: Filter Out 0s in a number list

- Create a list of numbers (with lots of 0's)
  - Ex: [0, 0, 0, 1, 2, 3, 4, 0, 0, 0, 5, 6]
- Use a Loop (your choice) to return a list without 0's
- Hint: to make a list with non-0s, use
  - A script variable
  - the add function

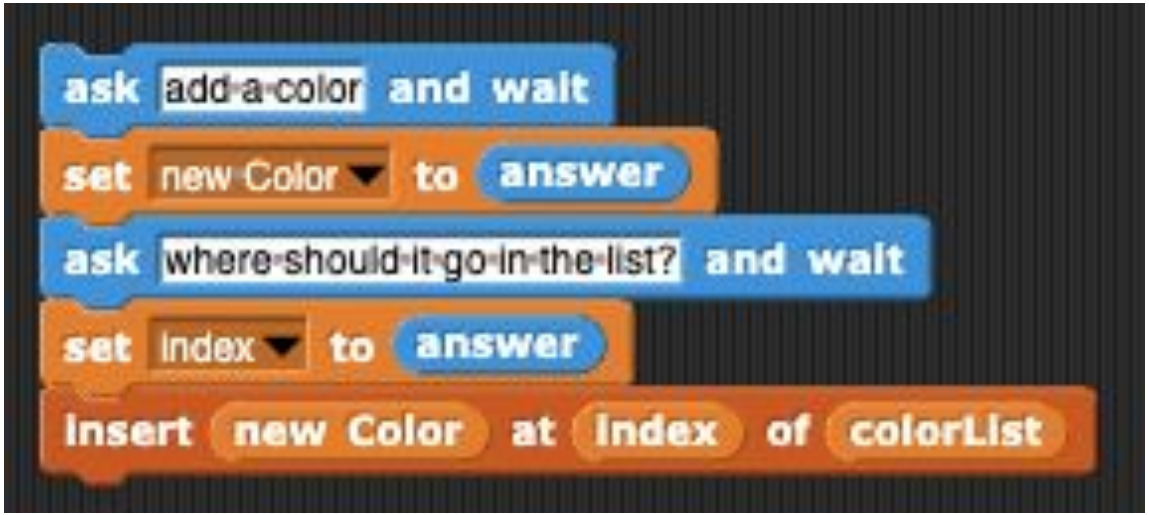
Lecture Guide: tinyurl.com/S24CS10L4

Functions: Saving Return Value in a Global Variable

Delete


ask delete a color (give the index number) and wait
set index to answer
delete index of colorList

Insert



- Adds element at a specific index position
- Pushes anything behind the new element back by 1 index position

Task: Use “Replace” to Pluralize Colors

- Instead of having `colors = [“Red”, “Blue”, “Green”]`
  I’d like to have `colors = [“Reds”, “Blues”, “Greens”]`

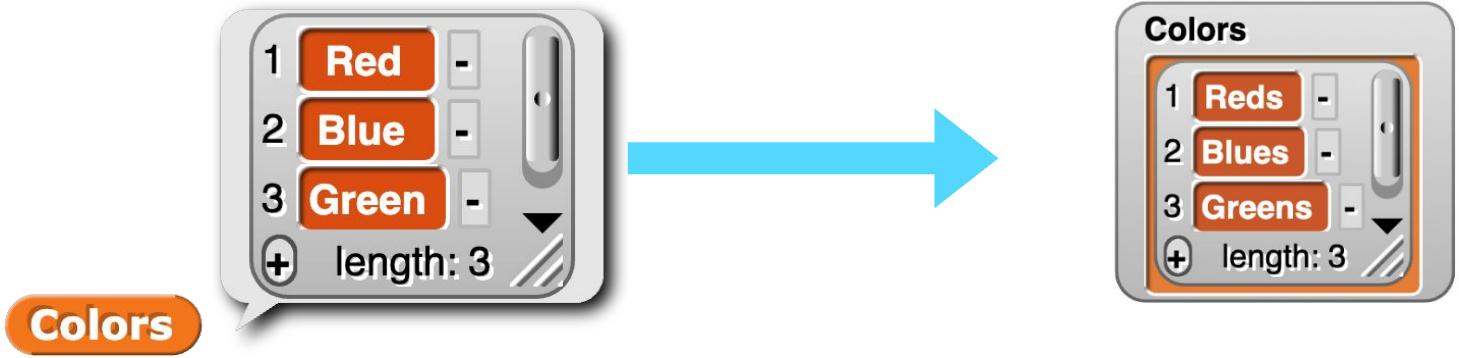
- How can we do this?

- Use a loop and the “replace” and “join”

Task: Pluralizing Colors

- Instead of having `colors = ["Red", "Blue", "Green"]`
  I'd like to have `colors = ["Reds", "Blues", "Greens"]`

```
for i = 1 to length of Colors
    replace item i of Colors with join item i of Colors s
```



TASK: PLURALIZING `COLORS`

- Notice: by **replacing** each original item with its plural form, I've actually changed the contents of the `Colors` list.
- Lists are **mutable**: the contents of a list can change (mutate) over time.
- Programs that alter the contents of a mutable object are called <u>destructive</u>.

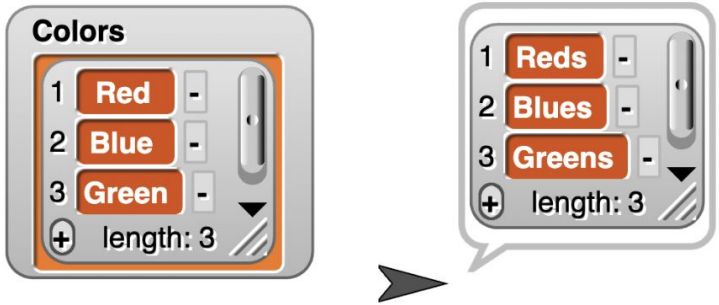
NON-DESTRUCTIVELY PLURALIZING `COLORS`

- What if instead of changing colors, I want to create a new list called `Plurals`, with the colors pluralized?

- Can use the `for each` block! -> return a different list

```
script variables Plurals
set Plurals to list
for each item in Colors
    add join item s to Plurals
say Plurals
```

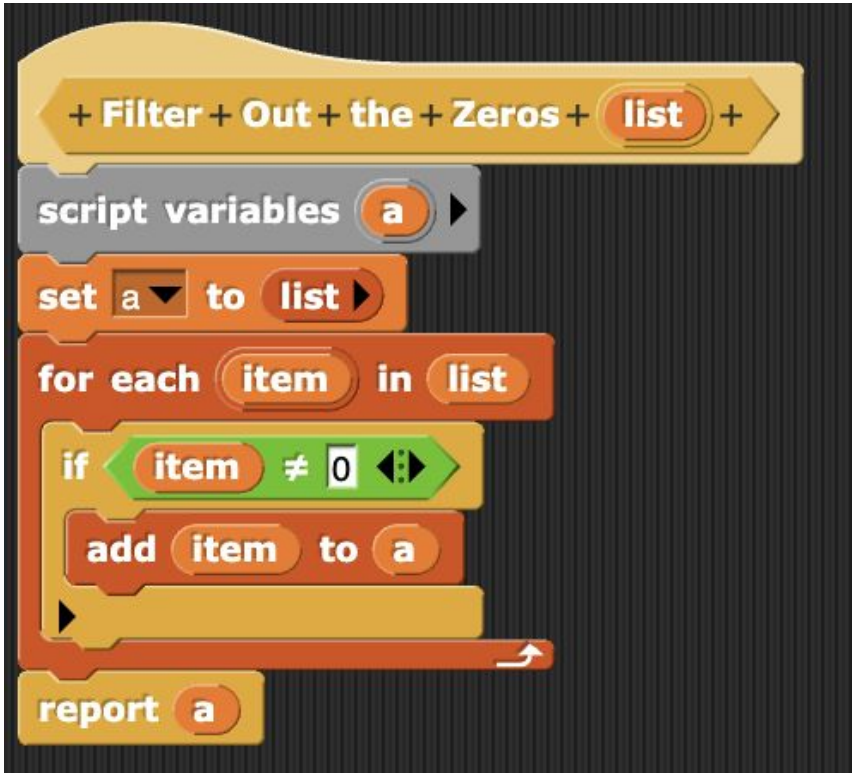
`Colors` remains unchanged!



Functions: Passing Lists as an argument

*Demo

Functions: Passing Lists as an argument



Functions: Saving Return Value in a Global Variable

Lab 4 Preview

- No Workbook
- 6 Challenges
  - Block 1: add all numbers from num1: _ to num2: _
  - Block 2: report only even numbers from list: _
  - Block 3: add start num: _ until > than stop num: _ and is odd
    - Cannot use “For Each” and “For i” blocks
  - Block 4: is num: _ prime?
  - Block 5: report only prime numbers from list: _
  - Block 6: report the duplicates in list: _

Task: prepend every list item with “good”

- Prepend = add to the front

Example 1: Access a list item using index

```
# Write a function named prepend_good that prepends
# the string “good ” to every element of the list.

# Note: You may assume that every element of the
# passed list is a string.

# Note: You may assume that your function will
# always be called with a list, however, you may not
# assume that that list is not empty.

# example input:
```

Further Exercises

If you don't feel comfortable with list methods or want to explore further, try out the exercises.

https://replit.com/@learntocodegpt/644-How-Many-Names

Solution: https://replit.com/@learntocodegpt/644-How-Many-Names-Solution

https://replit.com/@learntocodegpt/645-Five-Numbers

Solution: https://replit.com/@learntocodegpt/645-Five-Numbers-solution

https://replit.com/@blavde/847-Librarian#assignment.txt

Solution: https://replit.com/@learntocodegpt/647-Librarian-solution

https://replit.com/@learntocodegpt/6411-Take-a-Thing-Out-Sort-It-and-Reverse-It

Solution: https://replit.com/@learntocodegpt/6411-Take-a-Thing-Out-Sort-It-and-Reverse-It-Solution#main.py

https://replit.com/@learntocodegpt/6412-Librarian-Part-2

Solution: https://replit.com/@learntocodegpt/6412-Librarian-Part-2-Solution