Lesson 3
Lists

Accessing this Slide Deck

bit.ly/fdaPythonLists

LEARNING OBJECTIVES

You will be able to…

- **Store** multiple pieces of information within a list variable
- **Access** information stored within them using an index
- **Manipulate** a list using slicing and built-in functions

DO NOW

- Let’s come up with 5 examples of each of the following:
  - Items on a grocery list
  - Lists of cities in Ghana
  - Names of students
  - List of soccer teams
- A lot of things we encounter in life come in the form of lists of information!
  - Anybody have other examples?

REMINDER ABOUT VARIABLES

- Variables are **containers** that store numbers, strings, etc.

```
myFruit = "orange"
myGrade = 90
```

- What if I have multiple items I want to store?

REMINDER ABOUT VARIABLES

- We *could* create 4 separate variables:

```
1  favoriteFruit1 = "orange"
2  favoriteFruit2 = "pineapple"
3  favoriteFruit3 = "banana"
4  favoriteFruit4 = "mango"
```

But there is a better way!

LISTS

- Used to store multiple values (in one variable!)

```
favoriteFruits = ["orange", "pineapple", "banana", "mango"]
```

- Each item in the list is called a **member** (or **element**), and each member's position is called its **index**
- Lists are everywhere!

Interlude: Types in Python

Strings

- A data type in python
- We can create a **string** simply by enclosing characters in quotes
- Python treats single quotes (‘ ’) the same as double quotes (“ ”)
- A list of characters

- FOR EXAMPLE:
  - `stringVariable1 = 'Hello World!'`
  - `stringVariable2 = "Python Programming”`

Booleans

- Special reserved values that don’t need quotation marks
- Either **True** or **False**
- **NOT STRINGS -** different from “True” and “False”
- Use to make logical statements and comparisons

- FOR EXAMPLE:

```
print 5 > 3
print 5 < 3
print 3 * 2 > 4
```

Comparison Operator

- When I want to compare two things, I use the “Double Equals” symbol

- FOR EXAMPLE:

```
print “cat” == “dog”
print “dog” == “dog”
print 2 + 2 == 4
```

Integers

- A numerical data type
- They are often called just integers or **ints**
- Positive or negative whole numbers with no decimal point

- FOR EXAMPLE:
  - `integerVariable1 = 5`
  - `integerVariable2 = -76392`

Floating Point Values

- Another numerical data type, they are often just called **floats**
- They represent real numbers and are written with a **decimal point** dividing the whole and fractional parts
- **Note**: Any use of decimal point creates a float, which is a different type from an int!
- Division with floats returns exact numbers, while int division rounds down!

- FOR EXAMPLE:
  - `floatVariable1 = 5.0`
  - `floatVariable2 = -6.231`

Casting

You can convert between types by **casting**. You can cast one type to another by using a special keyword:

- Integer to string: str()

```
>> x = 10
>> str(x)
'10'
```

- String to integer: int()

```
>> y = "10"
>> int(y)
10
```

- This is necessary when printing values of multiple types:
  - `print "hello " + 5` → error!
  - `print "hello " + str(5)` → good!

Lists

Multiple-Type Lists

- Most of the time, it makes sense for all elements in the list to have the **same type** (think about a list of scores, or a list of names -- what types will these be?)

- However, we can also create a list with different types!

```
costs = ["5 dollars", 2, 3.0]
```

What types are all of these elements?

Lists Use Brackets

- The bracket operator [ ] always denotes a list

- Recall that functions use parentheses! ( )

- Creating a new (empty) list:

```
myFruits = []
```

Quick Review: INDEXING - refers to the position of an element within an ordered list

Indexing a list always starts at **ZERO**

```
favoriteFruits = ["orange", "pineapple", "banana", "mango"]
```

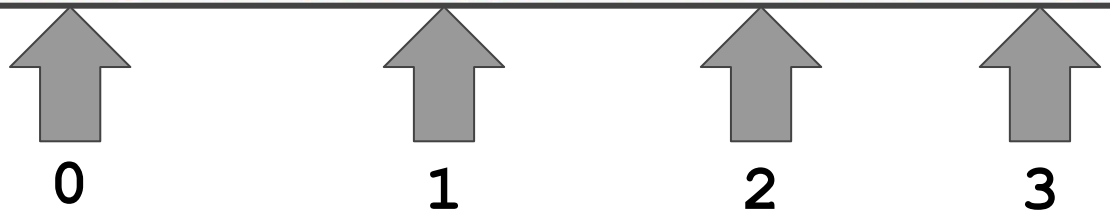
Learning Check!

What's the index of "orange" in our list favoriteFruits?

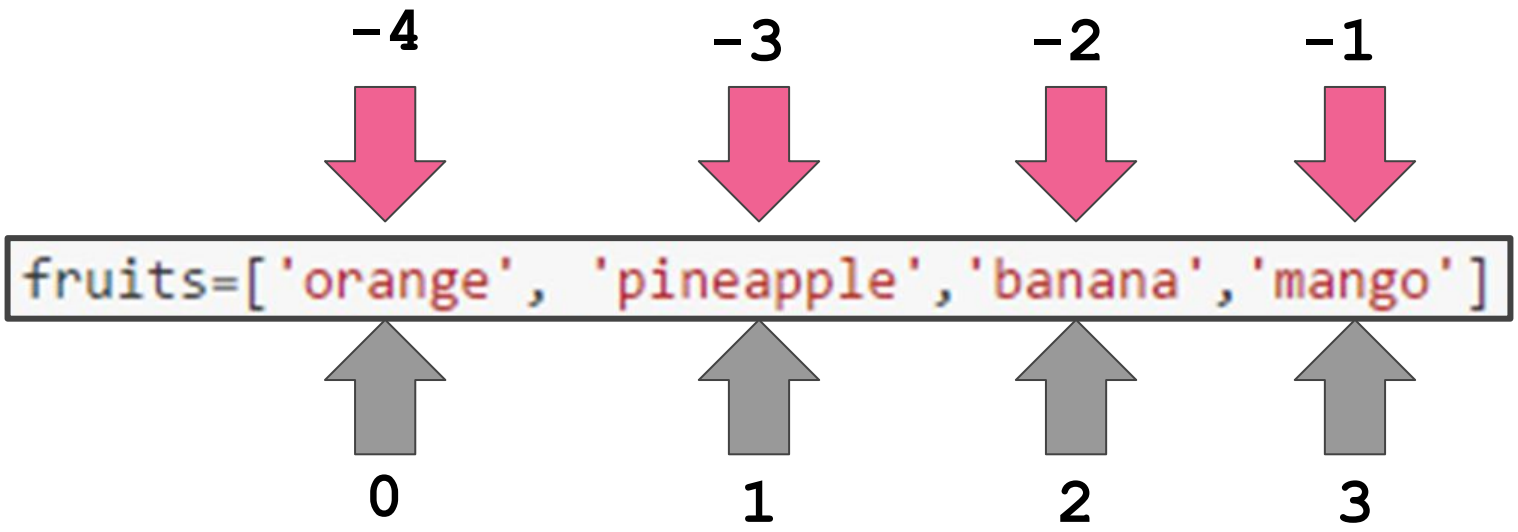
What about "banana"?

Learning Check Answer

```
fruits=['orange', 'pineapple','banana','mango']
```



You can also count through the list in **reverse**:



Slicing: You've already d☹ne it!

H	e	l	l	o	,		w	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12

```
message = "Hello, world!"
print message[0:6]
```

What will the **output** be?

Slicing: You've already d☹ne it!

H	e	l	l	o	,		w	o	r	l	d	!
0	1	2	3	4	5	6	7	8	9	10	11	12

```
message = "Hello, world!"
print message[0:6]
```

What will the **output** be? `"Hello,"`

INDEXING AND SLICING

```
squares = [1, 4, 9, 16, 25]
```

Indexing returns an **item**

```
squares[0] =
squares[1] =
squares[2] =

squares[-1] =
squares[-2] =
squares[-3] =
```

Slicing returns a **list**

```
squares[0:2] =
squares[1:3] =
squares[2:4] =

squares[:4] =
squares[3:] =
```

INDEXING AND SLICING

```
squares = [1, 4, 9, 16, 25]
```

Indexing returns an **item**

```
squares[0] = 1
squares[1] = 4
squares[2] = 9

squares[-1] =
squares[-2] =
squares[-3] =
```

Slicing returns a **list**

```
squares[0:2] =
squares[1:3] =
squares[2:4] =

squares[:4] =
squares[3:] =
```

INDEXING AND SLICING

```
squares = [1, 4, 9, 16, 25]
```

Indexing returns an **item**

```
squares[0] = 1
squares[1] = 4
squares[2] = 9

squares[-1] = 25
squares[-2] = 16
squares[-3] = 9
```

Slicing returns a **list**

```
squares[0:2] =
squares[1:3] =
squares[2:4] =

squares[:4] =
squares[3:] =
```

INDEXING AND SLICING

```
squares = [1, 4, 9, 16, 25]
```

Indexing returns an **item**

```
squares[0] = 1
squares[1] = 4
squares[2] = 9

squares[-1] = 25
squares[-2] = 16
squares[-3] = 9
```

Slicing returns a **list**

```
squares[0:2] = [1,4]
squares[1:3] = [4,9]
squares[2:4] = [9,16]

squares[:4] =
squares[3:] =
```

INDEXING AND SLICING

```
squares = [1, 4, 9, 16, 25]
```

Indexing returns an **item**

```
squares[0] = 1
squares[1] = 4
squares[2] = 9

squares[-1] = 25
squares[-2] = 16
squares[-3] = 9
```

Slicing returns a **list**

```
squares[0:2] = [1,4]
squares[1:3] = [4,9]
squares[2:4] = [9,16]

squares[:4] = [1,4,9,16]
squares[3:] = [16,25]
```

INDEXING AND SLICING

Open brackets!

```
squares = [1, 4, 9, 16, 25]
```

```
print squares[0]
print squares[-1]
print squares[-3:]
```

1. Can you return a list of [1,4]?
2. What about [1, 9, 25]?

INDEXING AND SLICING

Open brackets!

```
squares = [1, 4, 9, 16, 25]
```

```
print squares[0]
print squares[-1]
print squares[-3:]
```

1. Can you return a list of [1,4]?
2. What about [1, 9, 25]?

```
answer = squares[0:2]
```

```
answer = [squares[0],squares[2], squares[4]]
```

FUNCTIONS TO HELP US USE LISTS

Open brackets!

- What do you think the following functions do?
  - `myList.append(x)`
  - `myList.extend(otherList)`

```
>>> myList = ["A","B","C","D"]
>>> myList.append("E")
>>> print myList
```

```
>>> otherList = ["F","G"]
>>> myList.extend(otherList)
>>> print myList
```

FUNCTIONS TO HELP US USE LISTS

Open brackets!

- What do you think the following functions do?
  - `myList.append(x)`
  - `myList.extend(otherList)`

```
>>> myList = ["A","B","C","D"]
>>> myList.append("E")
>>> print myList
['A', 'B', 'C', 'D', 'E']
>>> otherList = ["F","G"]
>>> myList.extend(otherList)
>>> print myList
```

FUNCTIONS TO HELP US USE LISTS

Open brackets!

- What do you think the following functions do?
  - `myList.append(x)`
  - `myList.extend(otherList)`

```
>>> myList = ["A","B","C","D"]
>>> myList.append("E")
>>> print myList
['A', 'B', 'C', 'D', 'E']
>>> otherList = ["F","G"]
>>> myList.extend(otherList)
>>> print myList
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

FUNCTIONS TO HELP US USE LISTS

Open brackets!

- What do you think the following functions do?
  - `myList.insert(i, x)`
  - `myList.remove(x)`

```
foods = ["kenkey", "waakye", "jollof"]
foods.insert(1, "fufu")
print foods
foods.remove("kenkey")
print foods
```

FUNCTIONS TO HELP US USE LISTS

Open brackets!

- What do you think the following functions do?
  - `myList.insert(i, x)`
  - `myList.remove(x)`

```
foods = ["kenkey", "waakye", "jollof"]
foods.insert(1, "fufu")
print foods
foods.remove("kenkey")
print foods
```

```
['kenkey', 'fufu', 'waakye', 'jollof']
```

FUNCTIONS TO HELP US USE LISTS

Open brackets!

- What do you think the following functions do?
  - `myList.insert(i, x)`
  - `myList.remove(x)`

```
foods = ["kenkey", "waakye", "jollof"]
foods.insert(1, "fufu")
print foods
foods.remove("kenkey")
print foods
```

```
['kenkey', 'fufu', 'waakye', 'jollof']
['fufu', 'waakye', 'jollof']
```

ADDING ELEMENTS TO A LIST

Open brackets!

- What does the following code snippet do?

```
>>> print myList
['A', 'B', 'C', 'D', 'E', 'F', 'G']
>>> myThings = []
>>> myThings += ['My ' + myList[0]]
>>> print myThings
```

ADDING ELEMENTS TO A LIST

Open brackets!

- What does the following code snippet do?

```
>>> print myList
['A', 'B', 'C', 'D', 'E', 'F', 'G']
>>> myThings = []
>>> myThings += ['My ' + myList[0]]
>>> print myThings
['My A']
```

INDIVIDUAL ACTIVITY

Take 5 minutes to complete the first activity on the worksheet.

If you finish early, discuss your answers with either a teacher or a classmate.

SAMPLE CODE

Open brackets!

Here we see more **built-in** functions specific to lists!

```
fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
```

count() is a function that returns the number of times an element appears in a list

reverse() inverts the order of the list

TRY IT OUT:

```
fruits.count('orange')
fruits.count('apple')
fruits.count('mango')
```

```
fruits.reverse()
print fruits
```

Try making use of all of these functions on your own!

REPLACING ELEMENTS

Open brackets!

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> print letters

>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>> print letters

>>> # now remove them
>>> letters[2:5] = []
>>> print letters

>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>> print letters
```

REPLACING ELEMENTS

Open brackets!

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>  print letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>>  print letters

>>> # now remove them
>>> letters[2:5] = []
>>>  print letters

>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>>  print letters
```

REPLACING ELEMENTS

Open brackets!

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>  print letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>>  print letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>>  print letters

>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>>  print letters
```

REPLACING ELEMENTS

Open brackets!

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>  print letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>>  print letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>>  print letters
['a', 'b', 'f', 'g']
>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>>  print letters
```

REPLACING ELEMENTS

Open brackets!

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>>  print letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> # replace some values
>>> letters[2:5] = ['C', 'D', 'E']
>>>  print letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']
>>> # now remove them
>>> letters[2:5] = []
>>>  print letters
['a', 'b', 'f', 'g']
>>> # clear the list by replacing all the elements with an empty list
>>> letters[:] = []
>>>  print letters
[]
```

CHALLENGE EXERCISE

- Remember functions from the last lesson!
- Create a **function** that takes in two lists. It removes the last element from the first list, reverses the second list, and creates one big list combining the two!
- Remember to always break down your tasks into manageable steps: **Step 1, Step 2, Step 3**

Solution to Challenge

Can get a solution in just one line!

```
def combine_lists(list1, list2):
    list2.reverse()
    return list1[:-1] + list2
```

Did we have any other creative solutions?

INDIVIDUAL ACTIVITY

Take 5 minutes to complete **Problem 2** on the worksheet.

If you finish early, discuss your answers with either a teacher or a classmate.

ASSESSMENT

SPEND ~10 MINUTES COMPLETING THE LAST TWO QUESTIONS ON THE WORKSHEET!

LET’S WRAP IT UP!

- What is a list? What do we use them for?
- What are a few different ways we can modify lists?
- What’s an example of a built-in Python function that works on lists?