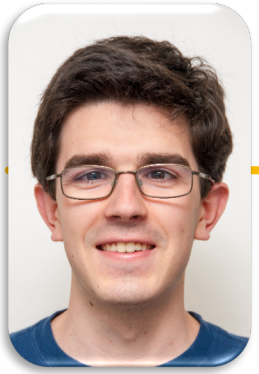

UC Berkeley EECS
Lecturer
Michael Ball

Lecture 4: Lists

News: "AI Algorithm Matches Cardiologists' Expertise, While Explaining Its Decisions"

- https://www.ucsf.edu/news/2021/08/421301/ai-algorithm-matches-cardiologists-expertise-while-explaining-its-decisions
- What are the most important findings of your study?
- [...] We developed an AI algorithm, called a "convolutional neural network," that outperformed a commonly used, commercial ECG analysis algorithm, and was comparable to expert cardiologists for many diagnoses.
- How did the algorithm do?
- [...] In addition, we used a technique that adds "explainability" to the algorithm, allowing it to highlight segments of the ECG that are important for each diagnosis that it makes. This provides doctors with important context about why [...]

Announcements

This week:

- Everyone should be getting off the waitlist.
  - Sign up for a section if you haven't!
  - https://sections.cs88.not.cs61a.org
- My OH regularly Weds 5-6pm in 625 Soda, except 9/22
- CS Mentors signups at 7pm today, see Ed.
  - 2 female identifying/nonbinary identifying sections!
    - Please use the signup form on Ed, since these are handled slightly differently.

Lists

Learning Objectives

- Lists are a new data type in Python.
- Lists can store any kind of data and be any length.
- We start counting items of lists at 0.
- Lists are *mutable*. We can change their data!

Lists

- A structure in Python that can hold many elements
  - Also referred to an an “array” in other programming languages.
- Lists are used to group similar items together.
  - A “contact list”, a “list of courses”, a “to do list”
- Python lists are *really* flexible!
  - Can contain any type of data
  - Can mix and match types!
  - Can add and delete items

Types We've Learned So Far

- Each *type* of data has a specific set of functions (methods) you can apply to them, and certain properties you can access.
- `int` / Integers
  - `1, -1, 0, …`
- `float` ("decimal numbers")
  - `1.0, 3.14159, 20.0`
- `string`
  - `"Hello, CS88"`
- `function`
  - `max(), min(), print(), your own functions!`
- **`list`**
  - **`['CS88', 'DATA8', 'POLSCI2', 'PHILR1B']`**

List Operations

- [] ”square brackets”: Used to access items in a list. We start at 0!
- len(): The number of items in a list
- +: We can add lists together
- min(), max(): Functions that take in a list and return some info.
- Converting between types: Strings and Lists:
  - <string>.split(<separator>) → List of string
    - “I am taking CS88.”.split(‘ ‘)
  - <string>.join(<list>) → String, with the items of a list joined together.
    - ” “.join([“I”, “am”, “taking”, “CS88.”])
- Lots more interesting tools!
  - https://docs.python.org/3.7/tutorial/datastructures.html

Selecting Elements From a List

- Selection refers to extracting elements by their index.
- Slicing refers to extracting subsequences.
- These work uniformly across sequence types.

```
L = [2,0,9,10,11]
S  =  "Hello, world!"
L[2]== 9
L[-1] == L[len(t)-1] == 11
S[1] == "e" # Each element of a string is a one-element string.
L[1:4] == (L[1], L[2], L[3]) == (0, 9, 10)
S[1:2] == S[1] == "e"
S[0:5] == "Hello", S[0:5:2] == "Hlo", S[4::-1] == "olleH"
```

Rules of Indexing & Slicing

- We start counting from 0.
  - You *will* mess this up. We all do. It's ok.
  - There's lots of bad dad jokes about this. ☺
- Python provides flexibility, but can be confusing.
  - `[0]` means the first item
  - `[-1]` means the last item, [-2] 2nd to last, and so on
- Slicing: The last value is *exclusive!*
  - `[:stop], e.g. my_list[:5] # items 0-4`
  - `[start:stop], e.g. my_list[2:5] # items 2,3,4`
  - `[start:stop:step] e.g. my_list[0:8:2] # items 0,2,4,6`

Demo

Sequences

Learning Objects

- Lists are a type of *sequence*
- *There are many types of sequences* in Python.
  - range
  - string
  - tuples
- Sequences all share some common properites.

Sequences

- The term sequence refers generally to a data structure consisting of an indexed collection of values, which we'll generally call elements.
  - That is, there is a first, second, third value (which CS types call #0, #1, #2, etc.)
- A sequence may be finite (with a length) or infinite.
- It may be mutable (elements can change) or immutable.
- It may be indexable: its elements may be accessed via selection by their indices.
- It may be iterable: its values may be accessed sequentially from first to last.

`range`

- range() is a built in Python tool that generates a sequence of numbers.
  - It does not return a list unless we explicitly ask for one.
- It has many options: start, stop, and step.
- Range is *lazy!* It can be iterated over, but doesn't compute all its values at once.
  - We'll revisit this later.
- **GOTCHA:** Range is exclusive in the last value!
  - **`range(10)` is a sequence on 10 numbers from 0 to 9.**
- https://docs.python.org/3.7/library/stdtypes.html?highlight=range#range

Tuples

- Tuples are represented by ( )
- They show up everywhere in Python, often implicitiy.
  - e.g. `a,b = 1, 2 # 1,2 is really (1,2)`
- Tuples are **immutable.**
  - **t[2] = 4 is an Error.**

`for` Loops

Learning Objectives: Using Lists in Practice

- `for` Loops are a "generic" way to iterate over data.
- Use `range` in a for loop

for statement – iteration control

- Repeat a block of statements for a structured sequence of variable bindings

```
<initialization statements>
for <variables> in <sequence expression>:
  <body statements>

<rest of the program>
```

while statement – iteration control

- Repeat a block of statements until a predicate expression is satisfied

```
<initialization statements>
while <predicate expression>:
    <body statements>

<rest of the program>


# Equivalent to a for loop:
index = 0
while index < len(my_list)
    item = my_list[index]
    …
    index += 1
```

List Comprehensions

Learning Objectives

- List comprehensions let us build lists "inline".
- List comprehensions are an *expression that returns a list*.
- We can easily "filter" the list using a conditional expression, i.e. `if`

Data-driven iteration

- describe an expression to perform on each item in a sequence
- let the data dictate the control
- In some ways, nothing more than a concise for loop.

```
[ <expr with loop var> for <loop var> in <sequence expr > ]
```

```
[ <expr with loop var> for <loop var> in <sequence expr >
if <conditional expression with loop var> ]
```

Control Structures Review

- **The result of** *list(range(0,10))* **is…**
-
- **A)** *[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]*
- **B)** *[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]*
- **C)** *[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]*
- **D)** *[1, 2, 3, 4, 5, 6, 7, 8, 9]*
- **E) an error**

- **http://bit.ly/88Lec3Q1**

Solution:
A) *list(range(m,n))* **creates a list with elements from m to n-1.**

iClicker Question

- What is the value of thing after running:
  - thing = [print('I like '+ course) for course in courses]

  - Nothing
  - ["I like CS88", "I like DATA8", …]
  - []
  - [None, None, None, None]
  - Error

Control Structures Review

The result of *len([i for i in range(1,10) if i % 2 == 0)])* **is…**

A) 5
B) 4
C) 3
D) 2
E) 1

Solution:
B) *len([2, 4, 6, 8]) == 4*

iClicker Question

```
>>> uni = 'The University of California at Berkeley'
>>> words = uni.split(' ')
>>> thing = [ w[0] for w in words ]
```

A) []
B) ['The', 'University', 'of', 'California', 'at', 'Berkeley']
C) 'TUoCaB'
D) ['T', 'U', 'o', 'C', 'a', 'B']
E) Error

Solution:
D)

Control Structures Review

- **The result of** *[i for i in range(3,9) if i % 2 == 1]* **is…**

A) *[3, 4, 5, 6, 7, 8, 9]*
B) *[3, 4, 5, 6, 7, 8]*
C) *[1, 3, 5, 7, 9]*
D) *[3, 5, 7, 9]*
E) *[3, 5, 7]*

-
- ***http://bit.ly/88Lec3Q2***

Solution:
E) *[3, 5, 7]*