

CSCI 127: Joy and Beauty of Data

Lecture 7: Lists

Reese Pearsall

Snowmester 2020

<https://reese.github.io/classes/127/main.html>

Announcements

Lab 4 and Program 1 due Thursday (Tomorrow) @ 11:59 PM

Program 2 due Wednesday 12/16 @ 11:59 PM

Today

Intro to list, Example programs using lists

Lists

A **list** is an ordered collection of items (elements)

Usually denoted by square brackets []

```
names = ["Reese", "Jane", "Jeff"]
```

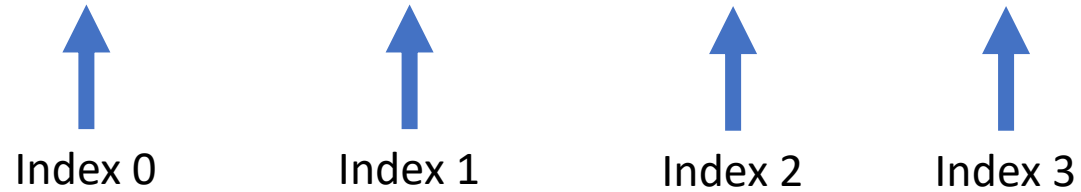
Can be of any data types

```
course = ["CSCI", 127, "Snowmester"]
```

Lists Index

List elements are ordered by their **index**

```
names = ["Reese", "Jane", "Jeff", "Tim"]
```



We can access specific elements of the list by providing the index of the element we want

<code>print(names[0])</code>	→	Reese	<code>print(names[-1])</code>	→	Tim
<code>print(names[1])</code>	→	Jane	<code>print(names[-2])</code>	→	Jeff
<code>print(names[3])</code>	→	Tim			
<code>print(names[4])</code>	→	Error (index 4 does not exist!)			

List Size

We can get the size/length of a list by using the **len()** function

```
names = ["Reese", "Jane", "Jeff", "Tim"]
```

```
print( len(names) ) → 4
```

```
primes =[2, 3, 5, 6, 7, 11, 13, 17, 19]
```

```
print( len(primes) ) → 9
```

List Size

We can get the size/length of a list by using the **len()** function

```
names = ["Reese", "Jane", "Jeff", "Tim"]
```

```
print( len(names) ) → 4
```

```
primes =[2, 3, 5, 6, 7, 11, 13, 17, 19]
```

```
print( len(primes) ) → 9
```

List Concatenation

We can concatenate lists together using the + operator

```
new_list = [1, 2, 3] + ["a", "b", "c"]  
print(new_list)
```

Output

```
[1, 2, 3, "a", "b", "c"]
```

List Repetition

We can repeat lists using the * operator

```
new_list = [1, 2, 3] * 3  
print(new_list)
```

Output

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```


List Membership

We can check to see if something is in a list using **in** and **not in**

```
cities = ["Billings", "Bozeman", "Butte", "Helena", "Missoula"]
```

```
print( "Billings" in cities )           True
```

```
print( "Red Lodge" in cities )         False
```

```
print( "bozeman" in cities )           False
```

```
print( "Butte" not in cities )         False
```

```
print( "Deer Lodge" not in cities )     True
```

List Mutability

Lists are **mutable** (we can change things in the list)

```
date = ["December", 9, 2020]
        0         1     2
```

```
print(date)
```

```
date[1] = 10
```

```
print(date)
```

```
date[2] = "2020"
```

```
print(date)
```

Output

```
["December", 9, 2020]
["December", 10, 2020]
["December", 10, "2020"]
```

List Slicing

We can access specific “slices” of lists using list slicing

0

1

2

3

4

```
cities = ["Billings", "Bozeman", "Butte", "Helena", "Missoula"]
```

```
cities[1:3] → ["Bozeman", "Butte"]
```

```
cities[0:4] → ["Billings", "Bozeman", "Butte", "Helena"]
```

```
cities[2:3] → ["Butte"]
```

```
cities[1:-1] → ["Bozeman", "Butte", "Helena"]
```

```
list_name[X:Y]
```

Get slice from X to Y but not including Y

Adding Element to list

We can add items to lists using **.append()** or **.insert()**

.append(*item*)

Add *item* to the end of the list

```
days = ["Mon", "Tue", "Wed"]
```

```
print(days)
```

```
days.append("Thurs")
```

```
print(days)
```

Output

```
["Mon", "Tue", "Wed"]  
["Mon", "Tue", "Wed", "Thurs"]
```

.insert(*position*, *item*)

Add *item* to specific index to list

```
days = ["Mon", "Tue", "Wed"]
```

```
print(days)
```

```
days.insert(1, "Thurs")
```

```
print(days)
```

Output

```
["Mon", "Tue", "Wed"]  
["Mon", "Thurs", "Tue", "Wed"]
```

Removing Items from Lists

We can remove items from lists using **.pop()** or **.remove()**

.pop(index) or .pop()

Removes item at specific index.
If index is not specified, then it will remove the last element in the list.

```
days = ["Mon", "Tue", "Wed"]
```

```
print(days)
```

```
days.pop(1)
```

```
print(days)
```

```
days.pop()
```

```
Print(days)
```

Output

```
["Mon", "Tue", "Wed"]
```

```
["Mon", "Wed"]
```

```
["Mon"]
```

.remove(item)

Will search through list for specific item and remove it

```
days = ["Mon", "Tue", "Wed"]
```

```
print(days)
```

```
days.remove("Tue")
```

```
print(days)
```

Output

```
["Mon", "Tue", "Wed"]
```

```
["Mon", "Wed"]
```

Counting frequency in lists

We can count the number of times an item appears in a list using **.count()**

```
scores = [100, 99, 81, 72, 100, 34, 86, 92, 100, 100, 66]
```

To print out the number of times **100** occurs in the list, we can do:

```
print(scores.count(100)) → 4
```

Sorting a List

We can sort elements in a list using `.sort()`

```
scores = [100, 99, 81, 72, 100, 34, 86, 92, 100, 100, 66]
```

```
print(scores)
```

```
scores.sort()
```

```
print(scores)
```

Output

```
[100, 99, 81, 72, 100, 34, 86, 92, 100, 100, 66]
```

```
[34, 66, 72, 81, 86, 92, 99, 100, 100, 100, 100]
```

Sorted least to greatest!

Sorting a List

We can sort elements in a list using `.sort()`

```
words = ["Cat", "Dog", "Apple", "Banana"]
```

```
print(words)
```

```
words.sort()
```

```
print(words)
```

Output

```
["Cat", "Dog", "Apple", "Banana"]
```

```
["Apple", "Banana", "Cat", "Dog"]
```

Sorted alphabetically!

Reversing a List

We can reverse a list using `.reverse()`

```
words = ["Cat", "Dog", "Apple", "Banana"]
```

```
print(words)
```

```
words.reverse()
```

```
print(words)
```

Output

```
["Cat", "Dog", "Apple", "Banana"]
```

```
["Banana", "Apple", "Dog", "Cat"]
```

List has been reversed!

Iterating through items in a list

We can iterate through a list using a for loop in two different ways

By Element:

```
words = ["Cat", "Dog", "Apple", "Banana"]  
for each_word in words:  
    print(words)
```

By Position/Index:

```
words = ["Cat", "Dog", "Apple", "Banana"]  
for i in range(len(words)):  
    print(words[i])
```

Example

Write a python function that will take in a list of test scores (of any size) and returns the average of those exam scores

Write a python function that will take in a list of words and return a list with all duplicate words removed

Write a python function that will return the largest value in a list. Next, generate a list of a random size (between 10 and 20) that is filled with random integers between 1 and 1000 and pass it into the function you created

Write a python function that will calculate the mode in a list of integers (you can always assume there will always be one single answer)

Announcements (Thursday)

Lab 4 and Program 1 due **TONIGHT** @ 11:59 PM

Lab 5 due Tuesday 12/15 @ 11:59 PM (files)

Program 2 due Wednesday 12/16 @ 11:59 PM

Today

More on lists, nested lists

Announcements (Thursday)

Lab 4 and Program 1 due **TONIGHT** @ 11:59 PM

Lab 5 due Tuesday 12/15 @ 11:59 PM (files)

Program 2 due Wednesday 12/16 @ 11:59 PM

Today

More on lists, nested lists



Warm up

The variable `scores` contains scores of each 2017 MSU football game.

Complete the program so that MSU'S wins and losses are computed, regardless of the number of games played

```
scores = [0, 31, 27, 31, 49, 21, 17, 25] #[msu-score-1, opponent-score-1, msu-score2,  
                                           # opponent-score2, ...]  
  
## The missing code goes here. Assume that every game results in either a win or loss  
  
print("MSU has", wins, "win(s) and", losses, "loss(es)")
```

Creating a list from a string

A list can be created from a string using `.split()`

```
declaration = "Four score and seven years ago"  
x = declaration.split()  
print(x)  
print(type(x))  
print(len(x))
```

Output

```
['Four', 'score', 'and', 'seven', 'years', 'ago']  
<class 'list'>  
6
```

```
declaration = "Four score and seven years ago"  
x = declaration.split("and")  
print(x)  
print(type(x))  
print(len(x))
```

Output

```
['Four score', 'seven years ago']  
<class 'list'>  
2
```

Creating a string from a list

A string can be created from a list using `.join()`

```
seuss = ["green", "eggs", "and", "ham"]
glue = "*"
glue = glue.join(seuss)
print(glue)
print(type(glue))
```

Output

```
'green*eggs*and*ham'
<class 'str'>
```

```
seuss = ["green", "eggs", "and", "ham"]
space = " "
space = space.join(seuss)
print(space)
print(type(space))
```

Output

```
'green eggs and ham'
<class 'str'>
```


Tuple

A tuple is like a list, except it is immutable (you cannot change the elements inside them)

```
today = ("Thursday", "December", 10)
print(type(today))
print(today[0])
today[0] = "Friday"
```

← Tuples use parenthesis, not square brackets

← Trying to change an element of a tuple results in an error

Output

```
'green*eggs*and*ham'
<class 'tuple'>
Thursday
```

```
TypeError: 'tuple' object does not support item
assignment
```

You can easily convert a tuple to be a list:

```
today = list(today)
```

Nested Lists

You can have lists inside of lists

```
teams = [ ["New England", "Patriots"], ["Miami", "Dolphins"], ... ]  
print(teams[0])  
print(teams[0][0])  
print(teams[0][1])  
print(teams[1])  
print(teams[1][1])
```

Output

```
['New England', 'Patriots']  
New England  
Patriots  
['Miami', 'Dolphins']  
Dolphins
```

Nested Lists

You can have lists inside of lists

Index 0 Index 1

```
teams = [ ["New England", "Patriots"], ["Miami", "Dolphins"], ... ]
print(teams[0])
print(teams[0][0])
print(teams[0][1])
print(teams[1])
print(teams[1][1])
```

Index 0 of Index 1
i.e teams[1][0]

Output

```
['New England', 'Patriots']
New England
Patriots
['Miami', 'Dolphins']
Dolphins
```

Nested List Example

Consider the nested list that contains 2013 census population information when the total population was **316,128,839**

```
populations =  
[["California", 38332521],  
 ["Texas", 26448193],  
 ["New York", 19651127],  
 ["Florida", 19552860],  
 ["Illinois", 12882135],  
 ["Pennsylvania", 12773801],  
 ["Ohio", 11570808],  
 ["Georgia", 9992167],  
 ["Michigan", 9895622],  
 ["North Carolina", 9848060]]
```

Write a function that calculates and returns the total population of the list

Write a function that also prints the percentage of the U.S. population that lives in each of the 10 most populated states

If time persists:

- modify the `populations` list to include information about whether each state is landlocked (`True` or `False`)
- Calculate and print the percentage of the population in the 10 most populated states that lives in a landlocked state