CSCI 127: Joy and Beauty of Data

Lecture 7: Lists

Reese Pearsall

Snowmester 2020

https://reesep.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

```
print( names[0] )   → Reese
print( names[1] )   → Jane
print( names[3] )   → Tim
print( names[4] )   → Error
```

(index 4 does not exist!)

```
print( names[-1] )  → Tim
print( names[-2] )  → Jeff
```

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)