Intro to Programming (No Prior Experience)

Advanced Lists & Dictionaries

CSCI-UA.0002-009
M/W 2:00-3:15
Professor: Emily Zhao

Poll Everywhere

https://pollev.com/emilyzhao

Advanced Lists

Programming Challenge

```
# function: shuffle_list
# input: a list
# processing: create a shuffled copy of the list
# output: return the shuffled

list a = [1,2,3,4,5]
answer = shuffle_list(a)
print(answer)

>>> [2, 5, 4, 1, 3]
```

Shuffling / Randomizing a List

You can shuffle (or randomize) the elements in a list by applying the following algorithm:

- Create an empty list
- Enter into a while loop
- Obtain a random number between 0 and the length of the list you wish to shuffle
- Place a copy of the data that exists at this random position into our new list
- Remove the item from original list
- Test the length of the original list – if it is zero we can end the while loop
- Your new list now contains a shuffled version of your original list

```
import random

# function: shuffle_list
# input: a list
# processing: create a shuffled copy of the list
# output: return the shuffled list

def shuffle_list(oldList):

    newList = []

    # while there are still items in the old list
    while len(oldList) > 0:

        #pick a new position and add it to the new list
        pos = random.randint(0, len(oldList)-1)
        newList.append( oldList[pos] )

        # remove element by position from old list
        del oldList[pos]

    return newList
```

Multi-dimensional lists

- All of the lists we have been creating so far have been one dimensional (i.e. linear) in nature
- You can create a two dimensional list in Python by simply nesting a list inside of another list.

```
mylist = [ [ 'a', 'b', 'c' ],
           [ 'd', 'e', 'f'] ]

print (mylist[0])
print (mylist[0][0])
print (mylist[1][1])

>> ['a', 'b', 'c']
>> a
>> e
```

Given the following list:

```
myList = [True, [0, 1, 2], "a", ["x","y","z"], "b",
          [3, 4, ["cat", "dog"]], "c"]
```

Write the line of code that will print the following:

- `[0,1,2]` `myList[1]`
- `"b"` `myList[4]`
- `"x"` `myList[3][0]`
- `"dog"` `myList[5][2][1]`

Dictionaries

Dictionaries

- A dictionary is a data structure for storing pairs of values
- Unlike a list, a Dictionary doesn’t use integer based index values.
- Instead, Dictionaries use immutable objects (like Strings) to index their content. These are also call keys. Keys are unique and cannot be repeated within a dictionary.
- Values can be accessed by their keys. Like lists, dictionaries are mutable.

Lists vs. Dictionaries

	Order	Access	When to use
List			
Dictionary			

Lists vs. Dictionaries

	Order	Access	When to use
List	ordered		
Dictionary	unordered		

Lists vs. Dictionaries

	Order	Access	When to use
List	ordered	numeric index	
Dictionary	unordered	immutable key (i.e. String)	

Lists vs. Dictionaries

	Order	Access	When to use
List	ordered	numeric index	when order matters
Dictionary	unordered	immutable key (i.e. String)	need to access values with a unique key

Trace the Output

```
myList = []
myDict = {}

myList[0] = "Emily"
print(myList)

myDict["Emily"] = 26
print(myDict)
```

Trace the Output

```
myList = []
myDict = {}

myList[0] = "Emily"
print(myList)

myDict["Emily"] = 26
print(myDict)
```

```
    myList[0] = "Emily"
IndexError: list assignment index out of range
```

```
{'Emily': 26}
```

Class Roster Demo (Lists + Dictionary)

Creating a Dictionary

- You can create a Dictionary using the curly braces – “{“ and “}”, like this:

```
my_dictionary = { }
```

- This will create an empty Dictionary

Adding items to a Dictionary

- In order to add an item to a Dictionary you need to specify a “key” – this is usually in the form of a String

- You can then associate some data with that key. For example I could associate the number 3.2 with the key “python” by doing this:

```
my_dictionary[“python”] = 3.2
```

- This will place the number 3.2 into the Dictionary at the position marked by the String “python”

Accessing Dictionary items

- You can access all items in a Dictionary by printing it out, like this:

```
print (my_dictionary)
```

- However, you often just want to access one item – this works the same as with an array, but you will use a key instead of an integer index:

```
print ( my_dictionary[“python”] )
```

Creating a Dictionary with Values

- Dictionaries store key / value pairs. You can initialize a Dictionary with a known set of key / value pairs by using the following syntax:

```
my_dictionary = { “python”:3.2, “java”:1.8 }
```

- This will create a Dictionary with the keys “python” and “java”

Invalid Indexes

- Note that you cannot access elements in a Dictionary that have not been defined. This would raise an exception if “java” was not a key in the Dictionary:

```
print ( my_dictionary[“java”] )
```

- You can use the “in” operator to test to see if a key is in a dictionary like this:

```
if ( “java” in my_dictionary” ):
```

- Note that this will check for the presence of a key in a dictionary, not for the data that the key is storing!

Deleting a key in a Dictionary

- You can use the del command to delete a key in a Dictionary, like this:

```
del my_dictionary[“java”]
```

- Make sure that you know that the key in question has been defined in the Dictionary before you run this command!

Clearing a Dictionary

You can clear all keys in a Dictionary by doing the following:

```
my_dictionary.clear()
```

Working with Dictionary Keys and Values

Cheat Sheet

myDictionary.keys()

myDictionary.values()

myDiction.items()

Iterating over every item in a Dictionary

- To iterate over every time in a Dictionary you need to be able to visit every key value (as opposed to simply knowing the size of a list and iterating over the integer range of the list)

- You can extract all the keys from a Dictionary by doing the following:
  `for key in my_dictionary.keys():`

- The target variable “key” will assume each key value in your Dictionary as the loop progresses.

- You can print out all items with their keys by doing the following:

```
for key in my_dictionary.keys():
  print (key, " == ", my_dictionary[key])
```

Iterating over every item in a Dictionary

- There is no guarantee that the keys() method will return the keys of a dictionary in any particular order.

- However, you can ask Python to sort the keys before you iterate over them, like this:

```
for key in sorted( my_dictionary.keys() ):
```

- This will sort the keys in ascending order, which then lets you access the elements in the dictionary in ascending order.

Finding all the values in a Dictionary

- You can also iterate over just the values in a dictionary (not just the keys) using this syntax:

```
for v in my_dictionary.values():
  print (v)
```

- Note that doing this will only expose the values in a dictionary and not the key – this means that you cannot change the values in the dictionary using this method. This is analogous to iterating over a list like this:

```
for item in my_list:
  print (item)
```

Iterating over every item in a Dictionary

- You can also iterate over a Dictionary by using the following technique to extract both the key and the value at the same time:

```
for key, value in my_dictionary.items():
  print (key, value)
```

Programming Challenges

How to import data from URL

```
import urllib.request

# define url
url = "http://www.example.com"

# initial request to URL
response = urllib.request.urlopen(url)

# read data from URL as string
data = response.read().decode("utf-8")
```

Programming Challenge

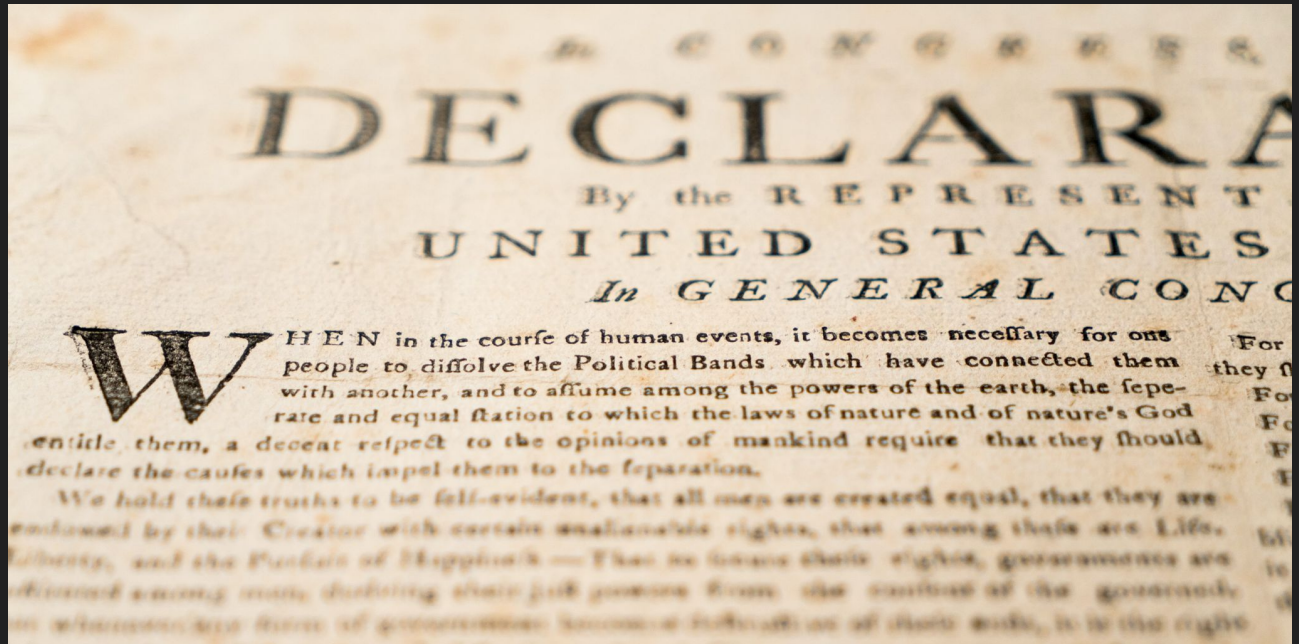
This data file contains all World Series winning teams up until a few years ago:

— http://002-text-files.glitch.me/world-series.txt

Write a program that reads in this data and finds the team that won the most games, now using a **dictionary**. Does it help?

Programming Challenge

- Count the frequency of all words in the Declaration of Independence.
- Which word is said the most?



Programming Challenge

Write a program that creates a dictionary containing the U.S. states as keys, and their capitals as values. Use this states.txt file to build your dictionary.

The program should then randomly quiz the user by displaying the name of a state and asking the user to enter that state's capital.

The user has 3 lives – meaning if they get three or more wrong answers, the game is over.