+ Review

www.pollev.com/python002

Programming Challenge

- Write a program that asks the user for a filename
- Then ask the user for a word to replace and its replacement
- Replace all occurrences of the word and write the result in a new file using the "_replaced" suffix (i.e. if the user enters the filename "test.txt" you would create a file called "test_replaced.txt"

Advanced Lists & Dictionaries

Introduction to Programming - Python

Advanced Lists

+

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

Programming Challenge

- Write a function according to the IPO Notation below:

```
# 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]
```

+
Programming Challenge: Poker

- Write a program that shuffles a deck of cards and selects five cards at random
- The list of cards is provided below.
- Shuffle the deck, deal four cards to the user, then ask if they would like to deal a second set of cards. Continue to deal the cards until you run out of cards in the deck.

```
cards = ['10 of Hearts', '9 of Hearts', '8 of Hearts', '7 of Hearts', '6 of Hearts', '5
of Hearts', '4 of Hearts', '3 of Hearts', '2 of Hearts', 'Ace of Hearts', 'King of
Hearts', 'Queen of Hearts', 'Jack of Hearts', '10 of Diamonds', '9 of Diamonds', '8 of
Diamonds', '7 of Diamonds', '6 of Diamonds', '5 of Diamonds', '4 of Diamonds', '3 of
Diamonds', '2 of Diamonds', 'Ace of Diamonds', 'King of Diamonds', 'Queen of Diamonds',
'Jack of Diamonds', '10 of Clubs', '9 of Clubs', '8 of Clubs', '7 of Clubs', '6 of
Clubs', '5 of Clubs', '4 of Clubs', '3 of Clubs', '2 of Clubs', 'Ace of Clubs', 'King
of Clubs', 'Queen of Clubs', 'Jack of Clubs', '10 of Spades', '9 of Spades', '8 of
Spades', '7 of Spades', '6 of Spades', '5 of Spades', '4 of Spades', '3 of Spades', '2
of Spades', 'Ace of Spades', 'King of Spades', 'Queen of Spades', 'Jack of Spades']
```

+ Multi Dimensional Lists

- All of the lists we have been creating so far have been one dimensional (i.e. linear) in nature
- High level programming languages also have the ability to construct lists that can store multiple values within the same element

+
Multi Dimensional Lists

- A one dimensional list can be thought of as a line
- A two dimensional list can be thought of as a plane
- A three dimensional list can be thought of as a cube
- … etc!

Creating a two dimensional list

- You can create a two dimensional list in Python by simply nesting a list inside of another list. Example:

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

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

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

Multi-Dimensional List Practice

- Given the following list,

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

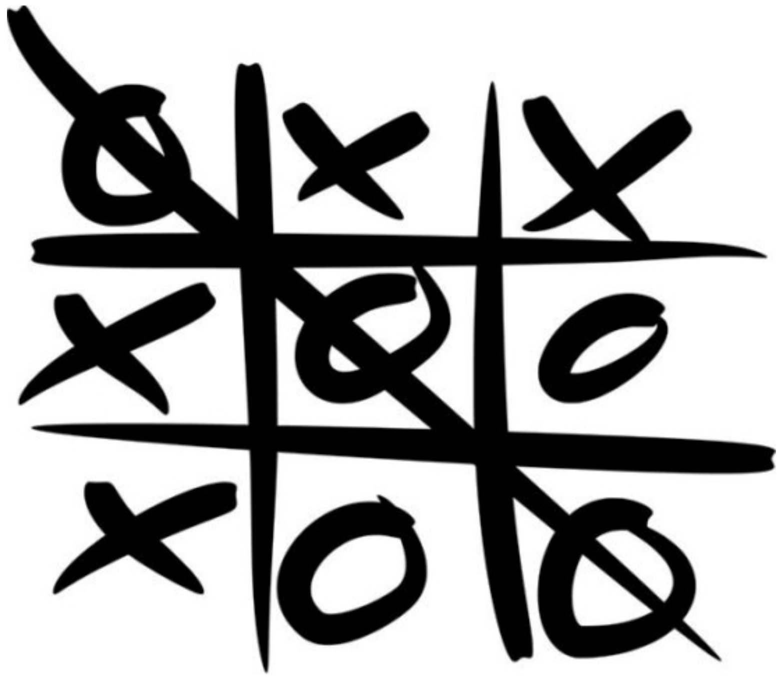
Write the line of code that will print the following:

- `[0,1,2]`
- “b”
- “x”
- “dog”

+

Programming Challenge: Tic Tac Toe

- Write a program that uses a multi dimensional list to simulate a tic tac toe board
- Initially set up your list to contain all "empty" spaces represented by a "." character
- Ask the user for a coordinate (i.e. 0, 0)
- If the space they request is still free you can place an X in the slot
- Have the computer pick a random spot to place an O character
- If there are no more spots left then you can end the game



+

Dictionaries

- A Dictionary in Python is a sequence object like a list
- Unlike a list, a Dictionary doesn’t use integer based index values.
- Instead, Dictionaries use immutable objects (like Strings) to index their content
- In other languages Dictionaries are often referred to as “Associative Arrays” or “Hashmaps”

Lists vs. Dictionaries

List	Dictionary
Sequence Structure	Sequence Structure
Mutable (can be changed if you know which index you are modifying)	Mutable (can be changed if you know which index you are modifying)
Items are stored in a particular order based on index values	Items are not stored in any particular order
Items can be indexed using an integer	Items can be indexed using anything that is immutable (integer, String, etc)

+

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”

+

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”] )
```

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!

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”]
  ```

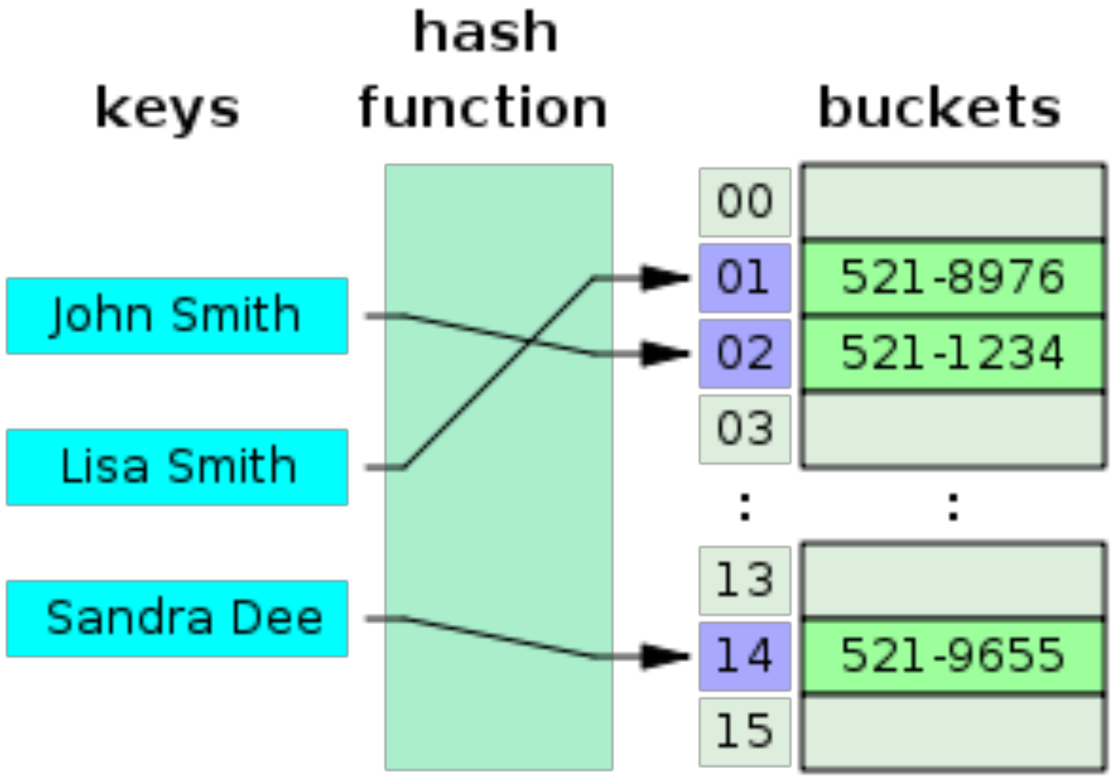
- Make sure that you know that the key in question has been defined in the Dictionary before you run this command!

Clearing a Dictionary

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

```
my_dictionary.clear()
```

+ Behind the Scenes: How Dictionaries work



Working with Dictionary Keys and Values

Finding all the keys in a Dictionary

- In a list we can easily visit all "slots" in the list by visiting every index value in the list, like this:

```
mylist = [ 'a', 'b', 'c', 'd' ]
for i in range(0, len(mylist)):
     print (mylist[i])
```

- However, we can't do this with a Dictionary since the index values for a dictionary are not necessarily going to be integers

- We can use the "keys()" method to ask a Dictionary object to expose all of the keys that are defined within that Dictionary like this:

```
my_dictionary.keys()
```

Iterating over every item in a Dictionary

- 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)
```

+
Which for loop should I use?

- Given the following dictionary,

```
myDictionary = {1:"January", 2:"February",
3:"March", 4:"April"}
```

- Print out the following:

```
January

February

March

April
```

Which for loop should I use?

- Given the following dictionary,

```
myDictionary = {1:"January", 2:"February",
3:"March", 4:"April"}
```

- Print out the following:

```
1 : January

2 : February

3 : March

4 : April
```

+

Programming Challenges

+
Programming Challenge

- Write a program that stores a series of price values in a dictionary along with the name of some products
- Here is an example dictionary:

```
groceries = {"apples":
2.25, "bananas": 1.25,
"cherries", 3.50}
```

- Then write a lookup program that lets the user enter in a product. If the product exists you can then display its price.

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 program should keep a count of the number of correct and incorrect responses.

+

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.

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.

+
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 program should ask the user 5 questions and keep a count of the number of correct and incorrect responses.