DICTIONARIES AND STRUCTURING DATA

CS 3080: Python Programming

UCCS University of Colorado
Colorado Springs

Question: Multiply each element of a list by a number

[1, 2, 3, 4, 5]   >> multiply by 5 >>   [5, 10, 15, 20, 25]

```
myList = [1, 2, 3, 4, 5]
myNewList = [i * 5 for i in myList]


print(myNewList)                # [5, 10, 15, 20, 25]
```

For example, in Matlab:
myList = myList .* 5

== vs is

```
>>> from copy import copy

>>> spam = [1, 2, 3]
>>> eggs = spam
>>> eggs == spam
True
>>> eggs is spam
True

>>> eggs = copy(spam)
>>> eggs == spam
True
>>> eggs is spam
False
```

is operator defines if both the variables point to the same object whereas **==** checks if the values for the two variables are the same

Dictionary

Also called associative arrays, maps, hashmaps, hashtables, etc.

- A **dictionary** is a collection which is unordered, changeable (**mutable**) and indexed.
  - *Typed with braces, { }*
  - *Use **keys** not **indexes***
    - Can use many different data types
    - Key with its associated value called *key-value pairs*.
    - You can access these values through their keys
    - Can use integers but do not have to start at zero
  - *Unordered*
    - No “first” item in a dictionary
    - Can’t be “spliced”
  - *KeyError for dictionary and IndexError for Lists.*

Dictionary methods

- .keys(), .values() and .items()
- Values returned by these methods are not true lists
  - *Cannot be modified*
  - *Do not have an append() method*
- The results of these methods are *dict_keys, dict_values,* and *dict_items* data types.
  - *It can be used in for loops!*
- To obtain a true list from one of these methods, pass its list-like return value to the list() function
- You can use the multiple assignment trick in a for loop to assign the key and value to separate variables when using .items().

in and not in

- Like lists, you can use in and not in operators to check if a key or value exists in a dictionary.

.get(‘key’, 0)

- Takes one or two arguments:
  - *the key of the value to retrieve. If key not found, the method will return* ***None****. >>> get(‘key’)*
  - ***Optional argument:*** *fallback value to return if that key does not exist. >>> get(‘key’, 0). If key not found, the method will return 0.*

myDict[‘key’]

- We can also access the value of a pair through myDict[‘key’]
  - *But if the key is not found, it will give us an error (note the difference between the .get(‘key’) method)*

.setdefault(‘key’,’value’)

- Can set a value in a dictionary for a certain key only if that key does not already have a value
- Code: .setdefault(‘key to check for’, ‘value to set if key does not exist)
- This method is a nice shortcut to ensure that a key exists.

Nested Dictionaries and Lists

- As you model more complicated things, you may find you need dictionaries and lists that contain other dictionaries and lists
- Lists are useful to contain an ordered series of values
- Dictionaries are useful for associating keys with values

Dict comprehensions

- As list comprehension, we can create dict comprehensions
- Curly braces for dicts, brakets for lists

Time to code – Number of occurrences – HW3 Ex 1

- Develop a program that counts the number of occurrences of each letter in a string. You can use the following string to test:
  - *It was a bright cold day in April, and the clocks were striking thirteen.*

- You may want to use the module pprint for pretty printing of dictionaries

```
import pprint

pprint.pprint(dictionary)
spam = pprint.pformat(dictionary) # Pretty text as a string value
print(spam)
```

Using Data Structures to Model Real-World Things

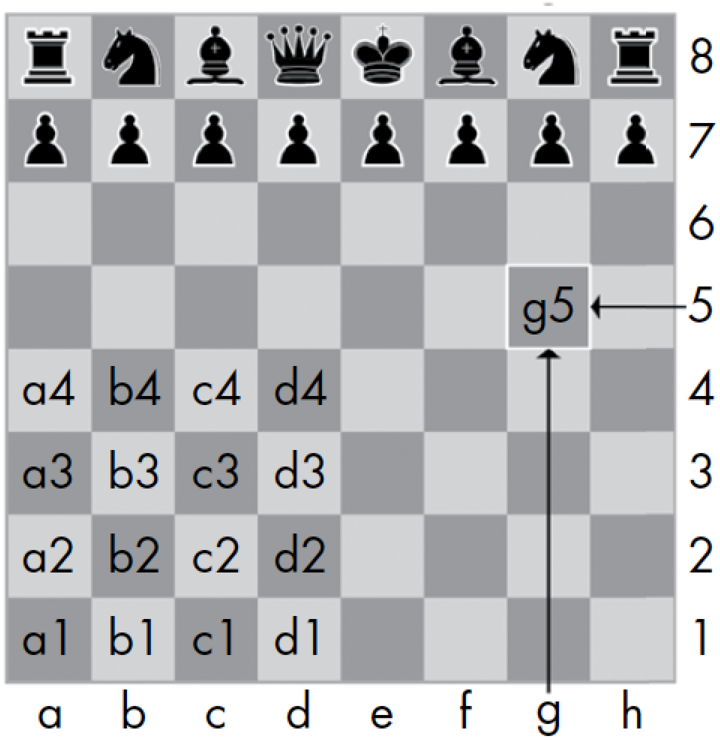
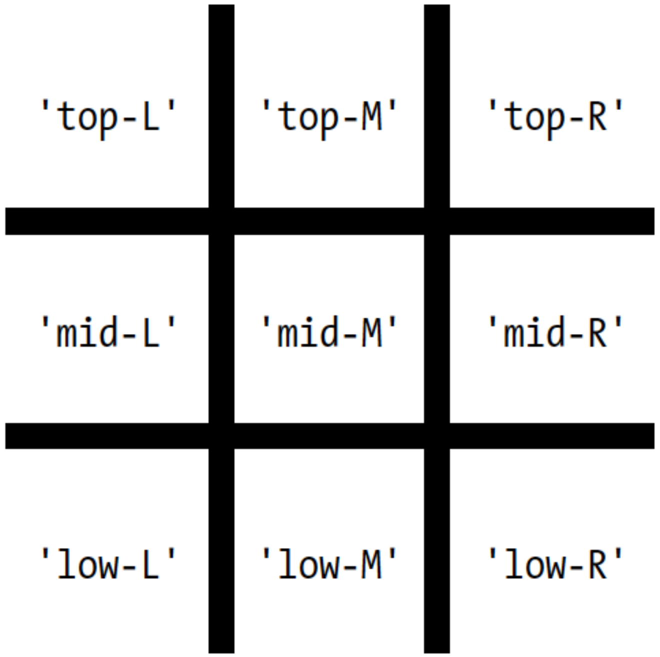


Figure 5-1: The coordinates of a chessboard in algebraic chess notation

- K for king, Q for queen, R for rook, B for bishop, and N for knight
- Describing a move uses the letter of the piece and the coordinates of its destination. A pair of these moves describes what happens in a single turn.
  - *Nf3 Nc6*
- A program on a modern computer can easily store billions of strings.
- This is how computers can play chess without having a physical chessboard. They model data to represent a chessboard, and you can write code to work with this model.
- This is where lists and dictionaries can come in. You can use them to model real-world things, like chessboards.



'top-L'	'top-M'	'top-R'
'mid-L'	'mid-M'	'mid-R'
'low-L'	'low-M'	'low-R'

Figure 5-2: The slots of a tic-tac-toe board with their corresponding keys

Time to code - Tic-Tac-Toe – HW3 Ex 2

- 1. Create the data structure
  - *Nine slots that can each contain an X, an O, or a blank.*
  - *To represent the board with a dictionary, you can assign each slot a string-value key.*
  - *String values to represent what's in each slot on the board: 'X', 'O', or ' '*
- 2. Create a function to print the board dictionary onto the screen
- 3. Add the code that allows the players to enter their moves

Time to code - Tic-Tac-Toe – HW3 Ex 2 – Output example

```
 | |
-+-+-
 | |
-+-+-
 | |
Turn for X. Move on which space?
top-L
X| |
-+-+-
 | |
-+-+-
 | |
Turn for O. Move on which space?
top-R
X| |O
-+-+-
 | |
-+-+-
 | |
```