Python Dictionaries

Chapter 9

Python for Everybody
www.py4e.com

What is a Collection?

- A collection is nice because we can put more than one value in it and carry them all around in one convenient package
- We have a bunch of values in a single “variable”
- We do this by having more than one place “in” the variable
- We have ways of finding the different places in the variable

What Is Not A "Collection"?

- Most of our variables have one value in them - when we put a new value in the variable - the old value is overwritten

```
$ python
>>> x = 2
>>> x = 4
>>> print(x)
4
```

A Story of Two Collections..

- List

  A linear collection of values
  Lookup by position 0 .. length-1

- Dictionary

  A linear collection of key-value pairs
  Lookup by "tag" or "key"

https://en.wikipedia.org/wiki/Index_card#/media/File:LA2-katalogkort.jpg
https://commons.wikimedia.org/wiki/File:Shelves-of-file-folders.jpg

Dictionaries (Part 1)

- Dictionaries are Python's most powerful data collection
- Dictionaries allow us to do fast database-like operations in Python
- Similar concepts in different programming languages
  - Associative Arrays - Perl / PHP
  - Properties or Map or HashMap - Java
  - Property Bag - C# / .Net

Dictionaries over time in Python

- Prior to Python 3.7 dictionaries did not keep entries in the order of insertion
- Python 3.7 (2018) and later dictionaries keep entries in the order they were inserted
- "insertion order" is not "always sorted order"

Below the Abstraction

- Python lists, dictionaries, and tuples are "abstract objects" designed to be easy to use
- For now we will just understand them and use them and thank the creators of Python for making them easy for us
- Using Python collections is easy. Creating the code to support them is tricky and uses Computer Science concepts like dynamic memory, arrays, linked lists, hash maps and trees.
- But that implementation detail is for another course…

Lists (Review)

- We append values to the end of a List and look them up by position
- We insert values into a Dictionary using a key and retrieve them using a key

```
>>> cards = list()
>>> cards.append(12)
>>> cards.append(3)
>>> cards.append(75)
>>> print(cards)
[12, 3, 75]
>>> print(cards[1])
3
>>> cards[1] = cards[1] + 2
>>> print(cards)
[12, 5, 75]
```

Dictionaries (Part 2)

- We append values to the end of a List and look them up by position
- We insert values into a Dictionary using a key and retrieve them using a key

```
>>> cabinet = dict()
>>> cabinet['summer'] = 12
>>> cabinet['fall'] = 3
>>> cabinet['spring'] = 75
>>> print(cabinet)
{'summer': 12, fall': 3, spring': 75}
>>> print(cabinet['fall'])
3
>>> cabinet['fall'] = cabinet['fall'] +
2
>>> print(cabinet)
{'summer': 12, 'fall': 5, 'spring': 75}
```

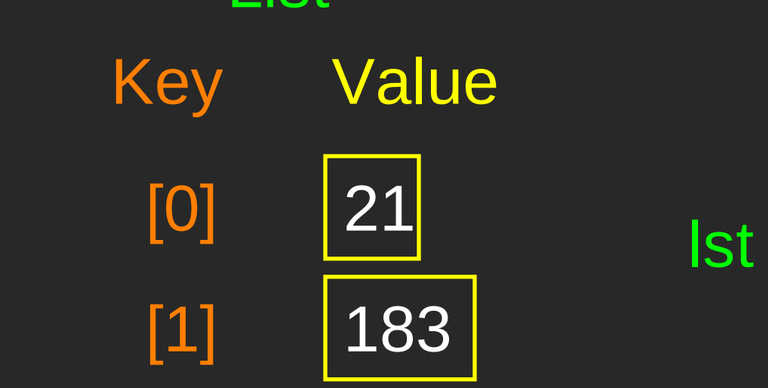
Comparing Lists and Dictionaries

Dictionaries are like lists except that they use keys instead of numbers to look up values

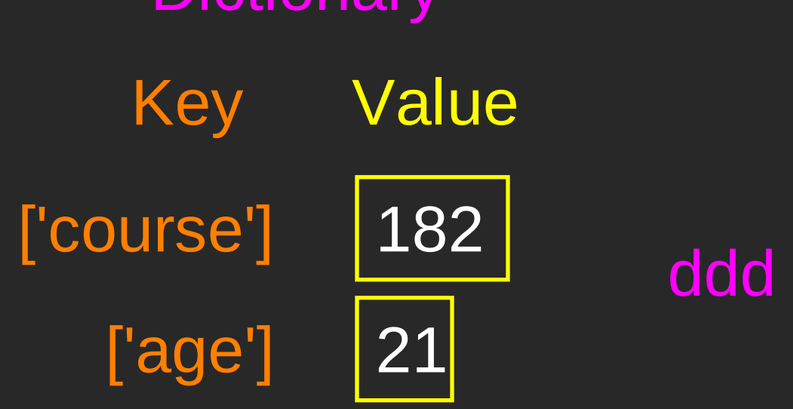
```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'age': 21, 'course': 182}
>>> ddd['age'] = 23
>>> print(ddd)
{'age': 23, 'course': 182}
```

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```




```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'age': 21, 'course': 182}
>>> ddd['age'] = 23
>>> print(ddd)
{'age': 23, 'course': 182}
```



```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```


List

Key	Value
[0]	23
[1]	183

lst

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'age': 21, 'course': 182}
>>> ddd['age'] = 23
>>> print(ddd)
{'age': 23, 'course': 182}
```

Dictionary

Key	Value
['course']	182
['age']	23

ddd

Dictionary Literals (Constants)

- Dictionary literals use curly braces and have a list of key : value pairs
- You can make an empty dictionary using empty curly braces

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(jjj)
{'chuck': 1, 'fred': 42, 'jan': 100}
>>> ooo = { }
>>> print(ooo)
{}
>>>
```