Dictionaries

A dictionary is a more general version of a list. Here is a list that contains the number of days in the months of the year:

```
days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

If we want the number of days in January, use days [0]. December is days[11] or days[-1]. Here is a dictionary of the days in the months of the year.

```
days = {'January':31, 'February':28, 'March':31, 'April':30,
        'May':31, 'June':30, 'July':31, 'August':31,
        'September':30, 'October':31, 'November':30, 'December':31}
```

The get the number of days in January, we use days ['January']. One benefit of using dictionaries here is the code is more readable, and we don't have to figure out which index in the list a given month is at. Dictionaries have a number of other uses, as well.

Basics

Creating dictionaries: here is a simple dictionary.

```
d = {'A':100, 'B':200}
```

To declare a dictionary, we enclose it in curly braces, {}. Each entry consists of a pair separated by a colon. The first part of the pair is called the key and the second is the value. The key acts like an index. So in the first pair, 'A':100, the key is 'A', the value is 100, and d['A'] gives 100. Keys are often strings, but they can be integers, floats, and may other

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

things as well. You can mix different types of keys in the same dictionary and different types of values, too.

Changing dictionaries: lets start with this dictionary.

```
d = {'A':100, 'B':200}
```

- To change d['A'] to 400, do

```
d['A']=400
```

- To add a new entry to the dictionary, we can just assign it, like below:

```
d['C']=500
```

- To delete an entry from a dictionary, use the del operator:

```
del d['A']
```

Empty dictionary: the empty dictionary is { }, which is the dictionary equivalent of [] for lists or ' ' for strings.

Important note: the order of items in a dictionary will not necessarily be the order in which put them into the dictionary. Internally, python rearranges things in a dictionary in order to optimize performance.

Working with Dictionaries

Copying dictionaries: to copy a dictionary, use its copy method. Here is an example:

```
d2 = d.copy()
```

in: the in operator is used to tell if something is a key in the dictionary. For instance, say we have the following dictionary:

```
d = {'A':100, 'B':200}
```

Referring to a key that is not in the dictionary will produce an error. For instance, print(d['c']) will fail. To prevent this error, we can use the in operator to check first if a key is in the dictionary before trying to use the key. Here is an example.

```
letter = input('Enter a letter: ')
if letter in d:
    print('The value is', d[letter])
else:
    print('Not in dictionary')
```

You can also use not in to see if a key is not in the dictionary.

Looping: looping through dictionaries is similar to loop through lists. Here is an example that prints the keys in a dictionary:

```
for key in d:
    print(key)
```

Here is an example that prints the values:

```
for key in d:
    print(d[key])
```

Lists of keys and values: the following table illustrates the ways to get list of keys and values from dictionary. It uses the dictionary d = { 'A':1,'B':3}.

Statement	Result	Description
`list(d)`	`['A','B']`	keys of d
`list(d.values())`	`[1,3]`	values of d
`list(d.items())`	`[('A',1),('B',3)]`	(key,value) pairs of d

The pairs returned by d.items are called tuples. Tuples are a lot like list. Here is a use of d.items to find all the keys in a dictionary d that correspond to a value of 100:

```
d = {'A':100, 'B':200, 'C':100}
L = [x[0] for x in d.items() if x[1]==100]
```

```
['A', 'C']
```

dict: the dict function is another way to create a dictionary. One use for it is kind of like the opposite of the items method:

```
d = dict([('A',100),('B',300)])
```

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

This creates the dictionary {'A':100, 'B':300}. This way of building a dictionary is useful if your program needs to construct a dictionary while it is running.

Dictionary comprehensions: dictionary comprehensions work similarly to list comprehensions. The following simple example creates a dictionary from a list of words, where the values are the lengths of the words.

```
d = {s : len(s) for s in words}
```

Exercises

1. Write a program that repeatedly asks the user to enter product names and prices. Store all of these in a dictionary whose keys are the product names and whose values are the prices. When the user is done entering products and prices, allow them to repeatedly enter a product name and print the corresponding price or a message if the product is not in the dictionary.
2. Using the dictionary created in the previous problem, allow the user to enter a dollar amount and print out all the products whose price is less than that amount.
3. For this problem, use the dictionary from the beginning of this lecture whose keys are month names and whose values are the number of days in the corresponding months.
    a) Ask the user to enter a month name and use the dictionary to tell them how many days are in the month.
    b) Print out all of the keys in alphabetical order.
    c) Print out all of the months with 31 days.
    d) Print out the (key-value) pairs sorted by the number of days in each month.
    e) Modify the program from part (a) and the dictionary so that the user does not have to know how to spell the month name

exactly. That is, all they have to do is spell the first three letters of the month name correctly.

4. Write a program that uses a dictionary that contains ten user names and passwords. The program should ask the user to enter their username and password. If the username is not in the dictionary, the program should indicate that the person is not a valid user of the system. If the username is in the dictionary, but the user does not enter the right password, the program should say that the password is invalid. If the password is correct, then the program should tell the user that they are now logged in to the system.
5. Repeatedly ask the user to enter a team name and the how many games the team won and how many they lost. Store this information in a dictionary where the keys are the team names and the values are lists of the form [wins, losses].
    a) Using the dictionary created above, allow the user to enter a team name and print out the team's winning percentage.
    b) Using the dictionary, create a list whose entries are the number of wins of each team. (c) Using the dictionary, create a list of all those teams that have winning records.
6. Repeatedly ask the user to enter game scores in a format like team1 score1 - team2 score2. Store this information in dictionary where the keys are the team names and the values are lists of the form [wins, losses].