5. Dictionaries and Structuring Data (Session)

09 March 2018 12:21

The Dictionary Data Type

```
>>> myCat = {'size': 'fat', 'color': 'gray', 'disposition': 'loud'}

>>> myCat['size']
'fat'
>>> 'My cat has ' + myCat['color'] + ' fur.'
'My cat has gray fur.'
```

Dictionaries vs. Lists

```
>>> spam = ['cats', 'dogs', 'moose']
>>> bacon = ['dogs', 'moose', 'cats']
>>> spam == bacon
False
>>> eggs = {'name': 'Zophie', 'species': 'cat', 'age': '8'}
>>> ham = {'species': 'cat', 'age': '8', 'name': 'Zophie'}
>>> eggs == ham
True
```

Because dictionaries are not ordered, they can't be sliced like lists.

You create an initial dictionary and store it in birthdays. You can see if the entered name exists as a key in the dictionary with the in keyword.

```
birthdays = {'Alice': 'Apr 1', 'Bob': 'Dec 12', 'Carol': 'Mar 4'}

if name in birthdays:
```

The keys(), values(), and items() Methods

There are three dictionary methods that will return list-like values of the dictionary's keys, values, or both keys and values: keys(), values(), and items(). The values returned by these methods are not true lists: They cannot be modified and do not have an append() method. But these data types (dict_keys, dict_values, and dict_items, respectively) can be used in for loops. To see how these methods work, enter the following into the interactive shell:

```
dict={'name':'c2t','age':10,5:100}

keys=dict.keys()
values=dict.values()
items=dict.items()

print('keys=',keys)
print('values=',values)
print('items=',items)

print('list(keys)=',list(keys))

for k,v in dict.items():
    print('k=',k,'v=',v)
```

The get() Method

```
>>> picnicItems = {'apples': 5, 'cups': 2}
>>> 'I am bringing ' + str(picnicItems.get('cups', 0)) + ' cups.'
'I am bringing 2 cups.'
>>> 'I am bringing ' + str(picnicItems.get('eggs', 0)) + ' eggs.'
'I am bringing 0 eggs.'
```