Dictionaries

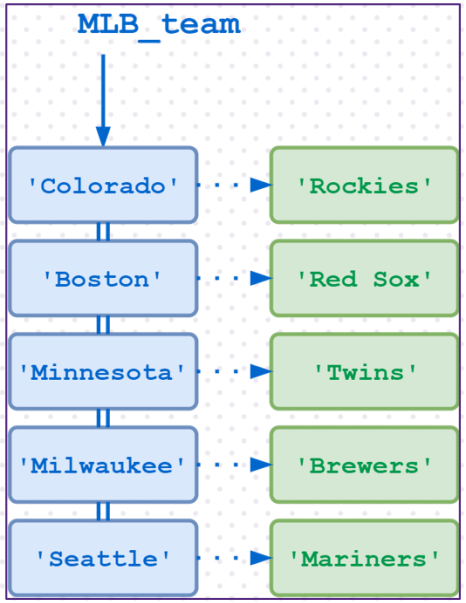
CSC 1200 - Principles of Computing

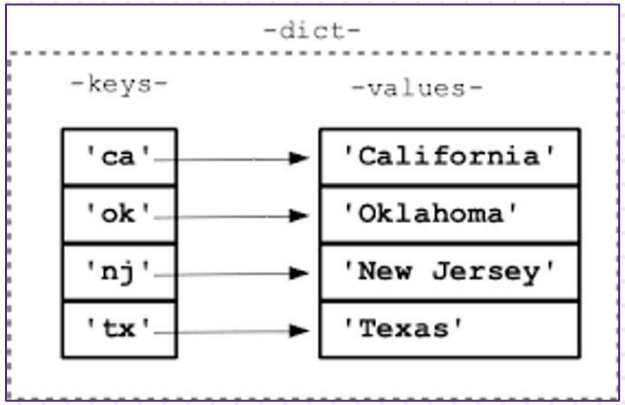
Overview

- Creating Dictionaries
- Using Keys to Index Dictionaries
- Functions and Operators for Dictionaries
- Dictionary Methods
  - The values Method
  - The keys Method
  - The get Method
- Dictionary Traversal
- Lookup and Reverse Lookup

Dictionary

- A dictionary is like a list, but more general.
  - In a list, the indices have to be integers (int)
  - In a dictionary, the indices can be almost any type
- A **dictionary** is a mapping between a set of indices, called **keys**, and a set of values.
- The association of a key and a value is called a **key-value pair** or an **item**.





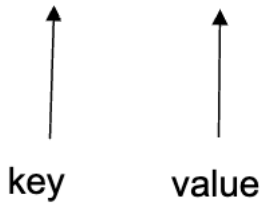
Creating a Dictionary in Python

- Recall that we specify a list using brackets []

```
>>> digit_list = [0,1,2,3,4,5,6,7,8,9]
>>> empty_list = []
>>> name_list = ['Amy', 'Beth', 'Carl', 'Doug']
```

- We specify a dictionary using curly braces (curly bois) {}

```
>>> digit_dict = { 'zero':0, 'one':1, 'two':2, 'three':3 }
>>> empty_dict = {}
>>> name_dict = { 'secretary':'Amy', 'treasurer':'Beth', 'president':'Carl', 'janitor':'Doug'}
```



Using Keys to Index Dictionaries

- We know that lists are indexed using integers 0,1,2,...
- Dictionaries are indexed using the keys.

```
>>> digit_dict['one']
1
>>> name_dict['president']
'Carl'
>>> name_dict['treasurer']
'Beth'
>>> name_dict['Amy']
Traceback (most recent call last):
  File "<pyshell#136>", line 1, in <module>
    name_dict['Amy']
KeyError: 'Amy'
```

The index MUST be a key. It cannot be a value.

Functions and Operators for Dictionaries

- The len function works on dictionaries. It returns the number of key-value pairs.

```
>>> print(engl_2_span)
{'one': 'uno', 'two': 'dos', 'three': 'tres', 'four': 'quatro', 'five': 'cinco', 'six': 'seis'}
>>> len(engl_2_span)
6
```

- The in operator tells you whether something appears as a *key* in the dictionary (not a value).

```
>>> 'four' in engl_2_span
True
>>> 'eight' in engl_2_span
False
>>> 'dos' in engl_2_span
False
```

The values Method

- To see whether something appears as a value in a dictionary, you can use the *values* method, which returns a list of values.

```
>>> engl_2_span.values()
dict_values(['uno', 'dos', 'tres', 'quatro', 'cinco', 'seis'])
>>> 'tres' in engl_2_span
False
>>> 'tres' in engl_2_span.values()
True
>>> 'four' in engl_2_span
True
>>> 'four' in engl_2_span.values()
False
```

The keys Method

- The *keys* method returns a list of keys in the dictionary.

```
>>> engl_2_span.keys()
dict_keys(['one', 'two', 'three', 'four', 'five', 'six'])
>>> 'four' in engl_2_span
True
>>> 'four' in engl_2_span.keys()
True
>>> 'quatro' in engl_2_span
False
>>> 'quatro' in engl_2_span.keys()
False
```

The get Method

- The *get* method takes a key and an optional second argument that is the default value. It returns the value associated with the *key* or the default value if the key is not in the dictionary.

```
>>> number = engl_2_span.get('three')
>>> number
'tres'
>>> number = engl_2_span.get('ten', 'Not Found')
>>> number
'Not Found'
>>> number = engl_2_span.get('nine')
>>> number
>>> print(number)
None
```

Dictionary Methods

Other dictionary methods include:

- pop - removes an element with a given key
- popitem - removes the last inserted key-value pair
- copy - returns a copy of the dictionary
- update - updates the dictionary with the specified key-value pair
- clear - removes all items from the dictionary

```
>>> engl_2_span
{'one': 'uno', 'two': 'dos', 'three': 'tres', 'four': 'quatro', 'five': 'cinco', 'six': 'seis'}
>>> engl_2_span.update({'zero':'cero'})
>>> engl_2_span
{'one': 'uno', 'two': 'dos', 'three': 'tres', 'four': 'quatro', 'five': 'cinco', 'six': 'seis',
 'zero': 'cero'}
>>> num = engl_2_span.pop('three')
>>> num
'tres'
>>> engl_2_span
{'one': 'uno', 'two': 'dos', 'four': 'quatro', 'five': 'cinco', 'six': 'seis', 'zero': 'cero'}
>>> numero = engl_2_span.popitem()
>>> numero
('zero', 'cero')
>>> engl_2_span
{'one': 'uno', 'two': 'dos', 'four': 'quatro', 'five': 'cinco', 'six': 'seis'}
```

Dictionary Methods (Continued)

```
>>> spanish_dict = engl_2_span
>>> english_dict = engl_2_span.copy()
>>> spanish_dict
{'one': 'uno', 'two': 'dos', 'four': 'quatro', 'five': 'cinco', 'six': 'seis'}
>>> english_dict
{'one': 'uno', 'two': 'dos', 'four': 'quatro', 'five': 'cinco', 'six': 'seis'}
>>> engl_2_span is spanish_dict
True
>>> engl_2_span is english_dict
False
>>> spanish_dict.clear()
>>> engl_2_span
{}
>>> spanish_dict
{}
>>> english_dict
{'one': 'uno', 'two': 'dos', 'four': 'quatro', 'five': 'cinco', 'six': 'seis'}
```

Dictionary Traversal

- If you use a dictionary in a for statement, it traverses the *keys* of the dictionary.

```
>>> for word in engl_2_span:
        print(word)


one
two
three
four
five
six
```

```
>>> for word in engl_2_span:
        print( word, '-->', engl_2_span[word])


one --> uno
two --> dos
three --> tres
four --> quatro
five --> cinco
six --> seis
```

Lookup and Reverse Lookup

- Given a dictionary d and a key k, it is easy to find the corresponding value v: v=d[k]
- Finding a value when given a key is called a **lookup**.
- A **reverse lookup**, is finding a key, when given a value. This is harder to do.
- Example: pet dictionary program (version 1 and 2)

Reverse Lookup Version 1

- This reverse lookup function returns the first index associated with the value or None if the value is not in the dictionary.

```
def reverse_lookup( dict, value ):
    for key in dict:
        if dict[key] == value:
            #value has been found, so return key
            return key
    #value was not assiciated with any key, so return None
    return None


############################
# Create the pet dictionary
############################

pets = { 'Trey':'fish',
         'Aiden':'dog',
         'Amelia':'cat',
         'Holly':'horse',
         'Anna':'dog',
         'Carly':'dog',
         'Daniel':'cat'
       }

for pet in pets.values():
    owner = reverse_lookup( pets, pet )
    print( pet, 'owned by', owner )

owner = reverse_lookup( pets, 'skunk' )
print( 'skunk owned by', owner )
```

```
fish owned by Trey
dog owned by Aiden
cat owned by Amelia
horse owned by Holly
dog owned by Aiden
dog owned by Aiden
cat owned by Amelia
skunk owned by None
```

Reverse Lookup Version 2

- This reverse lookup function returns a list of all the keys associated with the value.

```
def reverse_lookup( dict, value ):
    key_list = []
    for key in dict:
        if dict[key] == value:
            #value has been found, so add key to the list
            key_list.append(key)

    return key_list


############################
# Create the pet dictionary
############################

pets = { 'Trey':'fish',
         'Aiden':'dog',
         'Amelia':'cat',
         'Holly':'horse',
         'Anna':'dog',
         'Carly':'dog',
         'Daniel':'cat'
       }

for pet in pets.values():
    owner = reverse_lookup( pets, pet )
    print( pet, 'owned by', owner )

owner = reverse_lookup( pets, 'skunk' )
print( 'skunk owned by', owner )
```

```
fish owned by ['Trey']
dog owned by ['Aiden', 'Anna', 'Carly']
cat owned by ['Amelia', 'Daniel']
horse owned by ['Holly']
dog owned by ['Aiden', 'Anna', 'Carly']
dog owned by ['Aiden', 'Anna', 'Carly']
cat owned by ['Amelia', 'Daniel']
skunk owned by []
```

Dictionary Values Can Be Lists

- Lists can appear as values in a dictionary.

```
def invert_dict_lists(d):
    inverted = dict() #start with an empty dictionary
    for key in d:
        value = d[key]
        for v in value:
            if v not in inverted:
                #add v-key as a new item to the dictionary
                inverted[v] = [key]
            else:
                #append key to the existing list for v
                inverted[v].append(key)
    return inverted

#player_dict associates a name with a list of instruments played
player_dict = { 'Mateo': ['piano','guitar'],
                'Eddie': ['piano', 'saxophone'],
                'Trey': ['piano', 'drums'],
                'Asher': ['piano', 'violin', 'mandolin'],
                'Tia': ['piano', 'violin', 'cello', 'ukulele']
              }
#display the information in player_dict
for person in player_dict:
    print( person, 'plays', player_dict[person])

print('\n\n')

#create an inverted dictionary
instrument_dict = invert_dict_lists( player_dict )

for instrument in instrument_dict:
    print( instrument, 'played by', instrument_dict[instrument])
```

```
Mateo plays ['piano', 'guitar']
Eddie plays ['piano', 'saxophone']
Trey plays ['piano', 'drums']
Asher plays ['piano', 'violin', 'mandolin']
Tia plays ['piano', 'violin', 'cello', 'ukulele']



piano played by ['Mateo', 'Eddie', 'Trey', 'Asher', 'Tia']
guitar played by ['Mateo']
saxophone played by ['Eddie']
drums played by ['Trey']
violin played by ['Asher', 'Tia']
mandolin played by ['Asher']
cello played by ['Tia']
ukulele played by ['Tia']
```