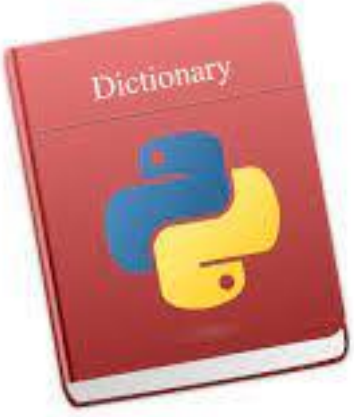
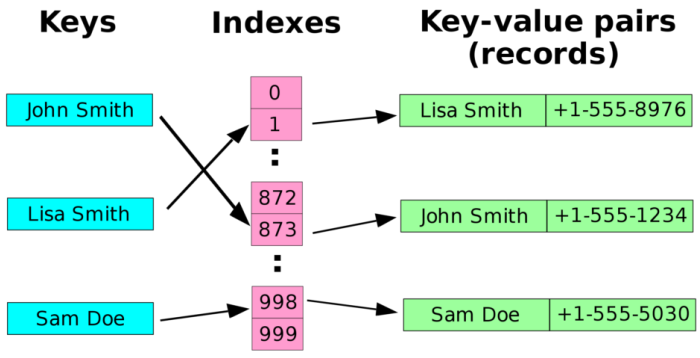
Chapter 9
Dictionaries

Outline

- What is a Dictionary?
- Basic Dictionary Operations
  - Creating a dictionary
  - Retrieval, modification, iterating over
- Dictionary Methods
- Duplicate Key Challenge
- More Complex Dictionaries
- Serializing Objects
- Dictionaries and JSON Files



What is a Dictionary?



- Dictionary is an unordered sequence of **key-value** pairs
  - Mutable, versatile, flexible
- Accessing a **dictionary value** through its **key** is much faster than accessing a **value** in a **list** through its **index**
  - Lists are sequential access data structures searched from beginning to end
  - Dictionaries are based on hash tables where an **index** of a **key** is computed based on hash (map) function – key hashing
  - Allows for (almost) direct access to the **value**

Basic Dictionary Operations

- **Lect9_Dictionaries.py**
- Creating a dictionary

```
>>> my_dict = {key1: value1, key2: value2, ...}
>>> cust_cities = {'Ryan': 'Santa Fe', ...}
```

  - Keys must be immutable objects: **strings**, **integers**, **tuples**
  - Values can be any object including **lists** and other **dicts**
- Retrieving a value from a dictionary

```
>>> my_dict[key1]  # Returns value1 or KeyError
>>> cust_cities['Ryan'] # Returns Santa Fe
```

  - Use **`in`** / **`not in`** operators to determine if the key exists
- Modifying dictionaries
  - Adding new **key: value** pairs, modifying or deleting existing ones
- Iterating over a dictionary
  - Using **for-loop** to move from one **key: value** pair to the next

Dictionary Methods

- Efficient methods for quick dictionary operations:
  - Access based on a key
    - **`get()`** method with default when key not found
  - Retrieval of all keys, values, key-value pairs
    - **`keys()`** method retrieves a tuple of all keys
    - **`values()`** method retrieves a tuple of all values
    - **`items()`** method retrieves a tuple of all key-value pairs
  - Removal from a dictionary
    - **`pop()`** method returns value based on a key and removes the key-value pair; default can be provided when key not found
    - **`popitem()`** method returns the last key-value pair before removing it

Duplicate Key Challenge

- Dictionary cannot have duplicate keys
  - Could use tuples with unique parts
  - Better to make sure all the keys are truly unique
- What if we want the same key to have multiple values?
  - Use **lists** to store multiple values for the same key

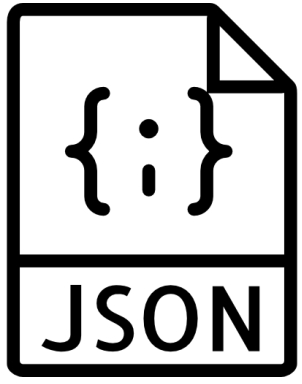
```
>>> my_dict = {key1: [val11, val12, val13],
               key2: [val21, val22], ...}
>>> cust_info = {'rmurphy1234': ['Ryan','Santa Fe'], ...}
```

  - The list values can be simple data types like strings and numbers or more complex objects like lists or dictionaries

More Complex Dictionaries

- Can build relatively complex data structures
  - **user_id** as the **key**
  - **List** of **values**
    - First element of the list is a simple string like **first name**
    - Second element of the list is a simple string like **city**
    - Third element of the list is yet another list (or a **sub-list**)
      - First element of the sub-list is a single key-value **mortgage** dictionary
      - Second element of the sub-list is a single element **car** dictionary, etc..
  - Recognize different data structures and their components
  - Conversion from complex data structure into CSV format

Dictionaries and JSON Files



- **Lect9_Dict_JSON.py**
- JSON = JavaScript Object Notation
  - Open-standard file format for sharing data
  - Used to send data between servers and Web browser clients
  - Self-documenting, relatively easy for humans to read
  - Consists of **attribute-value** pairs easily parsed
- Python's `json` package
  - Read data from JSON file with `load()` method
  - Analyze the resulting dictionary

Summary

- Defined dictionary
  - A sequence of key-value pairs
- Demonstrated basic dictionary operations
  - Creating, retrieving, adding, modifying, deleting
- Showcased the few important methods
  - Retrieving tuples of keys, values and key-value pairs
- Described the challenge of duplicate keys
  - Associate unique key with a list of (multiple) values
- Described more complex dictionaries
  - Allow us to design truly complex data structures
- Described the relationship between dicts and JSON files