PYTHON FOR DATA SCIENCE

UNIT 11

DICTIONARIES

MAIN TOPICS TO BE COVERED

- What dictionaries are
- Why dictionaries are used
- How to work with dictionaries
  - How to create dictionaries
  - How to add and remove dictionary items
- Dictionary methods

THE BASIC STRUCTURE

OF

DICTIONARIES

WHAT ARE DICTIONARIES?

- Dictionaries are one of Python's four basic built-in data structures (with the other three being lists, tuples, and sets).
- The dictionaries data structure holds **key-value pairs**.
- Dictionaries are similar to lists in that they hold values, but dictionaries use "keys" instead of numeric indexes.
- Other names for dictionaries … dict, hash, hash table
- Dictionaries are used to create a relationship between values and keys … i.e. we want to store a value and we want a "key" that allows us to access that value

HOW ARE DICTIONARIES CREATED?

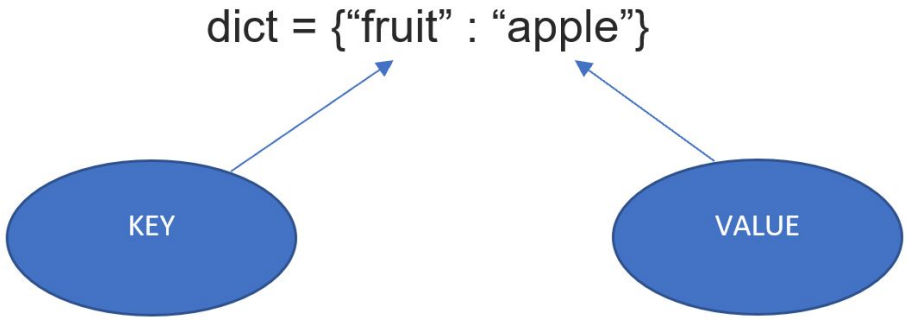
- You can define a dictionary by enclosing a comma-separated list of key-value pairs in curly braces (`{}`). A colon (`:`) separates each key from its associated value.

Python

```
d = {
    <key>: <value>,
    <key>: <value>,
      .
      .
      .
    <key>: <value>
}
```

- The key-value pairs are sometimes called items.

EXAMPLES: HOW DICTIONARIES ARE CREATED?

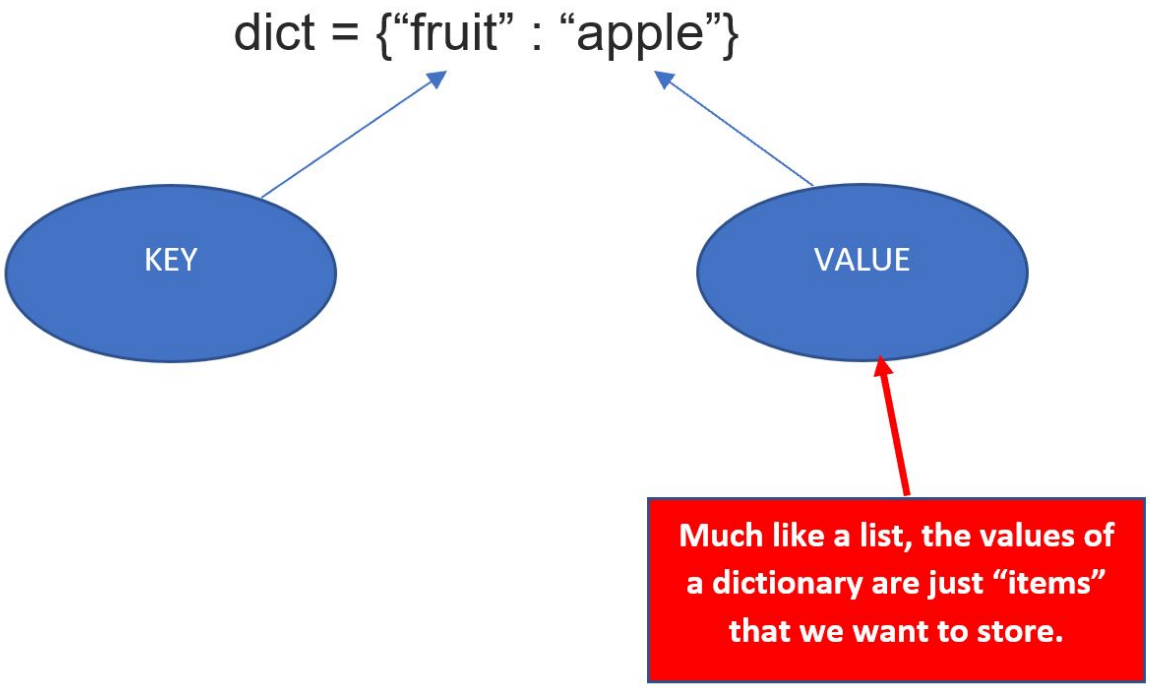


grades = {'John' : 'A', 'Emily' : 'A+', 'Betty' : 'B', 'Mike' : 'C', 'Ashley' : 'A'}

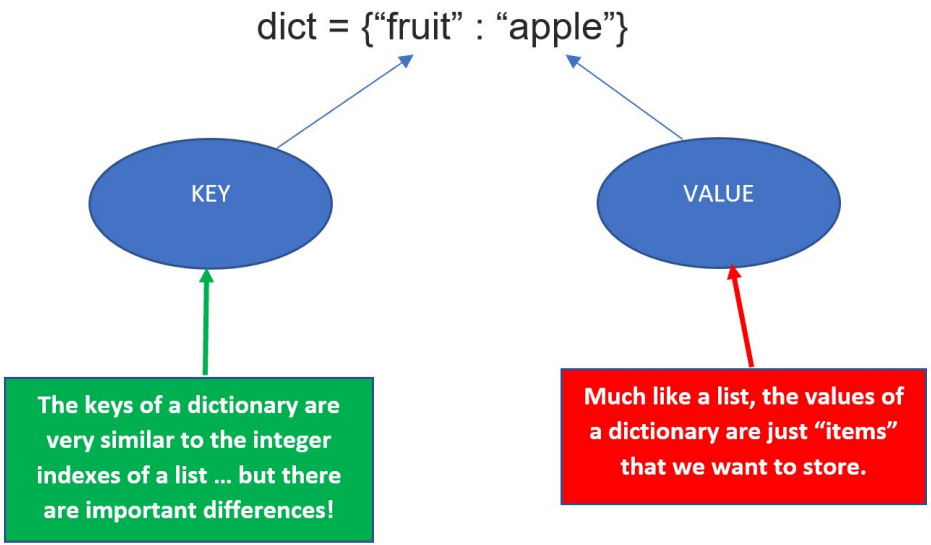
cardict = {"brand" : "Ford", "model" : "Mustang", "year" : 1964}

empty_dictionary = { }

DICTIONARY VALUES ARE LIKE THE VALUES OF A LIST



DICTIONARY KEYS CAN BE ANY IMMUTABLE OBJECT



- The keys of a dictionary can be any immutable object.
- So the keys of a dictionary can be integers, strings, or tuples … they cannot be lists.

RESTRICTIONS ON KEYS

- There can be no duplicate keys
  - Dictionary keys must be unique … i.e., no two values can have the same key
- Keys can be any immutable type
  - string
  - integer
  - boolean
  - float
  - tuple
  - (and other immutable types)

A SIMPLE EXAMPLE

OF A DICTIONARY

DICTIONARY EXAMPLE:

```
mlb_team = {   'Colorado' : 'Rockies'
             , 'Boston'   : 'Red Sox'
             , 'Minnesota': 'Twins'
             , 'Milwaukee': 'Brewers'
             , 'Seattle'  : 'Mariners'
           }
```

Here, we have created a dictionary with 5 key-value pairs.

DICTIONARY EXAMPLE:

This is a visual representation of the resulting dictionary.

```
mlb_team = {   'Colorado' : 'Rockies'
             , 'Boston'   : 'Red Sox'
             , 'Minnesota': 'Twins'
             , 'Milwaukee': 'Brewers'
             , 'Seattle'  : 'Mariners'
           }
```

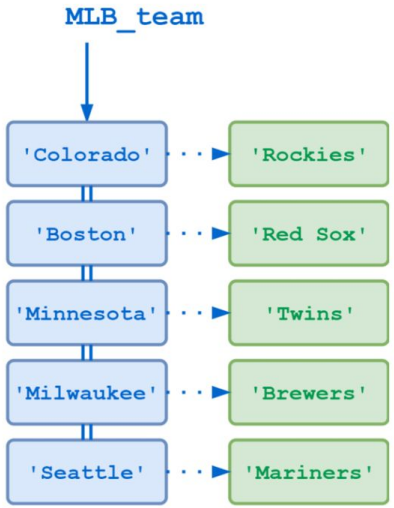
Key	Value
Colorado	Rockies
Boston	Red Sox
Minnesota	Twins
Milwaukee	Brewers
Seattle	Mariners

- There are 5 unique keys … these are the unique “parking spots” of the dictionary data structure.
- Every key has an associated value … i.e., for every parking spot, there is a car “parked” in that spot.

DICTIONARY EXAMPLE:

The dictionary mlb_team maps a location to the name of its corresponding Major League Baseball team:

```
mlb_team = {   'Colorado' : 'Rockies'
             , 'Boston'   : 'Red Sox'
             , 'Minnesota': 'Twins'
             , 'Milwaukee': 'Brewers'
             , 'Seattle'  : 'Mariners'
           }
```



Dictionary Mapping Location to MLB Team

GETTING VALUES

FROM A DICTIONARY

DICTIONARY INDEXING BASICS

- Use “bracket notation” to retrieve values … this is very similar to how we retrieve items / values from other data structures
- Dictionaries are indexed by *key* … this is different from how we index strings, lists, etc.
- To retrieve a specific value, you need to provide the key

SYNTAX:
HOW TO RETRIEVE A SINGLE ITEM FROM A DICTIONARY

- To retrieve an item from a dictionary, first type the name of the dictionary (dict), followed by brackets …

```
your_dictionary [ key-to-retrieve ]
```

- Inside of the brackets, type the key associated with the value you want to retrieve

SYNTAX:
HOW TO RETRIEVE A SINGLE ITEM FROM A DICTIONARY

```
mlb_team = {   'Colorado' : 'Rockies'
             , 'Boston'   : 'Red Sox'
             , 'Minnesota': 'Twins'
             , 'Milwaukee': 'Brewers'
             , 'Seattle'  : 'Mariners'
           }

mlb_team['Boston']
```

```
'Red Sox'
```

Key	Value
Colorado	Rockies
Boston	Red Sox
Minnesota	Twins
Milwaukee	Brewers
Seattle	Mariners

ADDING AND REMOVING

VALUES FROM A DICTIONARY

ADDING AND REMOVING VALUES

- Add values using “bracket” notation
  - Provide the key in brackets
  - Use the equal sign (=) to assign a value
- Delete values using the del operator

DELETE ITEMS WITH THE del Operator

```
mlb_team = {   'Colorado' : 'Rockies'
             , 'Boston'   : 'Red Sox'
             , 'Minnesota': 'Twins'
             , 'Milwaukee': 'Brewers'
             , 'Seattle'  : 'Mariners'
           }

print(mlb_team)

print()

del mlb_team['Minnesota']

print(mlb_team)
```

This code deletes the record associated with key ‘Minnesota’

```
{'Colorado': 'Rockies', 'Boston': 'Red Sox', 'Minnesota': 'Twins', 'Milwaukee': 'Brewers', 'Seattle': 'Mariners'}

{'Colorado': 'Rockies', 'Boston': 'Red Sox', 'Milwaukee': 'Brewers', 'Seattle': 'Mariners'}
```

Key	Value
Colorado	Rockies
Boston	Red Sox
Milwaukee	Brewers
Seattle	Mariners

ADD AN ITEM BY PROVIDING A NEW KEY IN BRACKETS, AND A VALUE

```
mlb_team = {   'Colorado' : 'Rockies'
             , 'Boston'   : 'Red Sox'
             , 'Minnesota': 'Twins'
             , 'Milwaukee': 'Brewers'
             , 'Seattle'  : 'Mariners'
           }

print(mlb_team)

print()

mlb_team['New York'] = 'Yankeees'

print(mlb_team)
```

This code adds the new value ‘Yankees’ which is associated with key ‘New York’

```
{'Colorado': 'Rockies', 'Boston': 'Red Sox', 'Minnesota': 'Twins', 'Milwaukee': 'Brewers', 'Seattle': 'Mariners'}

{'Colorado': 'Rockies', 'Boston': 'Red Sox', 'Minnesota': 'Twins', 'Milwaukee': 'Brewers', 'Seattle': 'Mariners', 'New York':
'Yankeees'}
```

Key	Value
Colorado	Rockies
Boston	Red Sox
Minnesota	Twins
Milwaukee	Brewers
Seattle	Mariners
New York	Yankees

SOME IMPORTANT

DICTIONARY METHODS

IMPORTANT DICTIONARY METHODS

- Dictionaries have several useful methods for retrieving data and for performing dictionary-manipulation.
- These methods use “dot” notation after the name of the dictionary.

```
name_of_dictionary.name_of_method
```

Method	What it does
`keys()`	retrieve all keys
`values()`	retrieve all values
`items()`	retrieve all items (key-value pairs returned as tuples)
`clear()`	delete all items from dictionary

WHEN TO USE

DICTIONARIES

WHEN TO USE DICTIONARIES

- When you want to store things that are “paired” together …
  - e.g., when two things are paired to one another
  - e.g., state abbreviation <---> state name
- When you need to retrieve data based on an “ID”, identifier, or key
  - When one item is used to look up another item

WE WILL GO OVER DICTIONARIES IN MUCH GREATER DETAIL IN THE JUPYTER NOTEBOOK PRESENTATION ON DICTIONARIES!!!