Data Structures

Chapter 12.3

Goals

- define data structure
- matrix example
- dictionary example
- explore code

Data Structure

Definition

- a data type that holds data in a convenient way to solve a problem
- sometimes built-in
- sometimes bespoke

Example

- Built-in **list** has methods append(), insert(), in
- Bespoke class **Person**() held attributes (Person.name, Person.email etc) and methods (Person.create_list(), Person._ _ repr _ _ ())

Matrix

Matrix

- Bespoke data structure with indexable rows and columns
- first index should be **row**
- second index should be **column**

$$\begin{array}{c|cccc|}
 & 1 & 2 & \cdots & n \\
1 & a_{11} & a_{12} & \cdots & a_{1n} \\
2 & a_{21} & a_{22} & \cdots & a_{2n} \\
3 & a_{31} & a_{32} & \cdots & a_{3n} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
m & a_{m1} & a_{m2} & \cdots & a_{mn}
\end{array}$$

Matrix

In Python

- each row can be a list
- each column is an item in the list

$$\begin{array}{c|cccc|}
 & 1 & 2 & \cdots & n \\
1 & a_{11} & a_{12} & \dots & a_{1n} \\
2 & a_{21} & a_{22} & \dots & a_{2n} \\
3 & a_{31} & a_{32} & \dots & a_{3n} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
m & a_{m1} & a_{m2} & \dots & a_{mn}
\end{array}$$

```
[[94, 94, 44, -73, -86, 46, 40, -88, -89, -36],
 [97, 45, -83, -68, 4, 72, 29, 21, 51, 3],
 [96, 71, 41, -96, 93, 55, -10, -12, 36, -37],
 [-87, 59, 6, 20, 77, -10, -92, 62, 14, 17],
 [-11, -55, -30, -60, -36, 22, 15, 20, -63, 43],
 [74, 81, 43, 30, 41, 40, -67, -23, -57, -27],
 [41, 18, -56, -100, -48, 14, -26, -29, -52, 11],
 [95, -11, -47, -50, 17, 64, 58, -67, 7, -55],
 [96, -82, 14, 100, -53, -93, 52, -95, -6, 34],
 [98, 41, 8, 94, 71, -57, 95, 46, -10, 13]]
```

Matrix

```python
import random

matrix = []
for i in range(10):
  matrix.append([])
  for j in range(10):
    matrix[i].append(random.randint(-100,100))
```

```
[[94, 94, 44, -73, -86, 46, 40, -88, -89, -36],
 [97, 45, -83, -68, 4, 72, 29, 21, 51, 3],
 [96, 71, 41, -96, 93, 55, -10, -12, 36, -37],
 [-87, 59, 6, 20, 77, -10, -92, 62, 14, 17],
 [-11, -55, -30, -60, -36, 22, 15, 20, -63, 43],
 [74, 81, 43, 30, 41, 40, -67, -23, -57, -27],
 [41, 18, -56, -100, -48, 14, -26, -29, -52, 11],
 [95, -11, -47, -50, 17, 64, 58, -67, 7, -55],
 [96, -82, 14, 100, -53, -93, 52, -95, -6, 34],
 [98, 41, 8, 94, 71, -57, 95, 46, -10, 13]]
```

Matrix

Indexing

```
for i in range(len(matrix)):
  for j in range(len(matrix[i])):
    # do something with matrix[i][j]
```

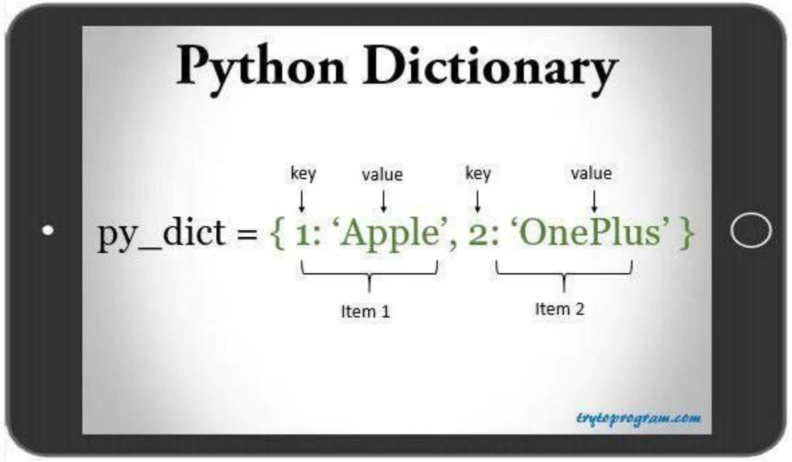
```
[[94, 94, 44, -73, -86, 46, 40, -88, -89, -36],
 [97, 45, -83, -68, 4, 72, 29, 21, 51, 3],
 [96, 71, 41, -96, 93, 55, -10, -12, 36, -37],
 [-87, 59, 6, 20, 77, -10, -92, 62, 14, 17],
 [-11, -55, -30, -60, -36, 22, 15, 20, -63, 43],
 [74, 81, 43, 30, 41, 40, -67, -23, -57, -27],
 [41, 18, -56, -100, -48, 14, -26, -29, -52, 11],
 [95, -11, -47, -50, 17, 64, 58, -67, 7, -55],
 [96, -82, 14, 100, -53, -93, 52, -95, -6, 34],
 [98, 41, 8, 94, 71, -57, 95, 46, -10, 13]]
```

Explore Matrix Code

Dictionary

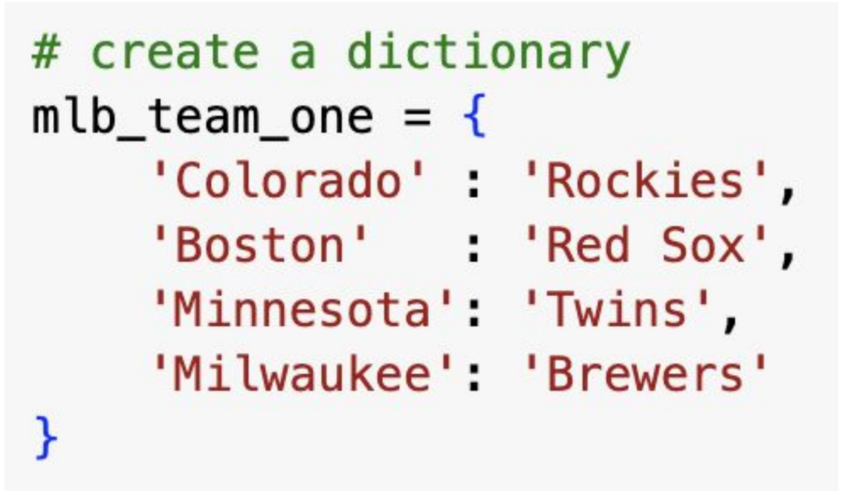
Dictionary

- Built-in data structure with keys and values
- every key is unique
- values are associated with keys, i.e. a key **points** to values
- there can be many key: value pairs inside one dictionary



Dictionary

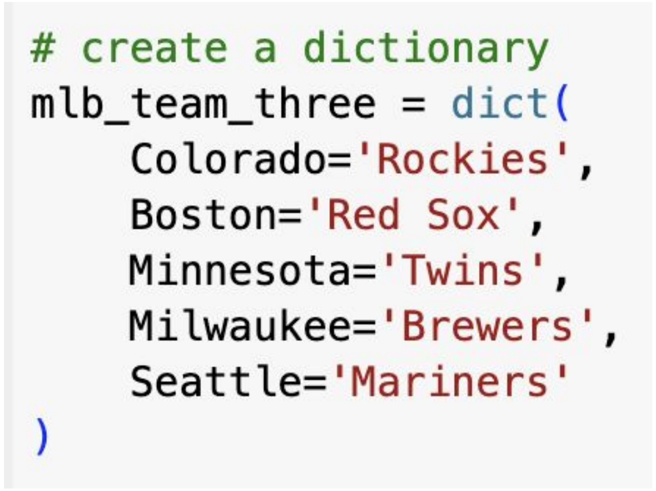
- Syntax for the container is {}
- keys appears first
- value(s) appear after the :
- key: value pairs are separated by ,



```python
# create a dictionary
mlb_team_one = {
    'Colorado' : 'Rockies',
    'Boston'   : 'Red Sox',
    'Minnesota': 'Twins',
    'Milwaukee': 'Brewers'
}
```

Dictionary

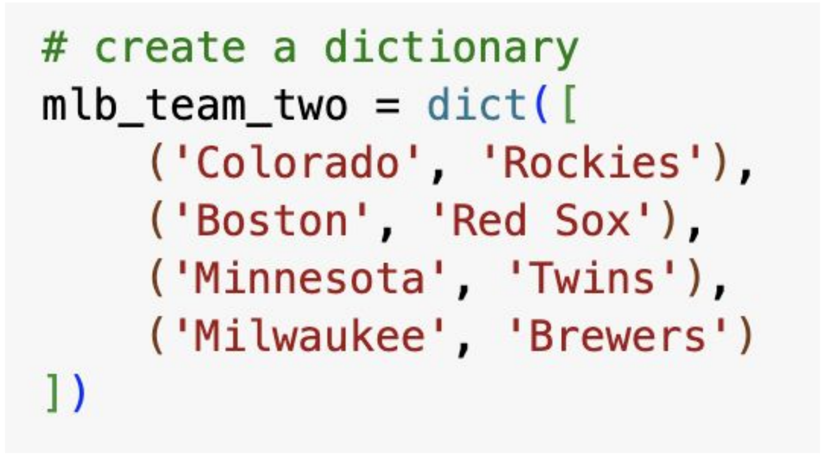
- Alternate syntax is using the type name **dict**
- keys appears first
- value(s) appear after the =
- key: value pairs are separated by ,



```
# create a dictionary
mlb_team_three = dict(
    Colorado='Rockies',
    Boston='Red Sox',
    Minnesota='Twins',
    Milwaukee='Brewers',
    Seattle='Mariners'
)
```

Dictionary

- Alternate syntax is using the type name **dict**
- key, value pairs are inside tuples, inside a list



```
# create a dictionary
mlb_team_two = dict([
    ('Colorado', 'Rockies'),
    ('Boston', 'Red Sox'),
    ('Minnesota', 'Twins'),
    ('Milwaukee', 'Brewers')
])
```

Dictionary

- values are accessed using the key

```
# create a dictionary
mlb_team_one = {
    'Colorado' : 'Rockies',
    'Boston'   : 'Red Sox',
    'Minnesota': 'Twins',
    'Milwaukee': 'Brewers'
}
```

```
# lookup specific values using a key
print(mlb_team_one['Minnesota'])
print(mlb_team_one['Colorado'])
```

Dictionary

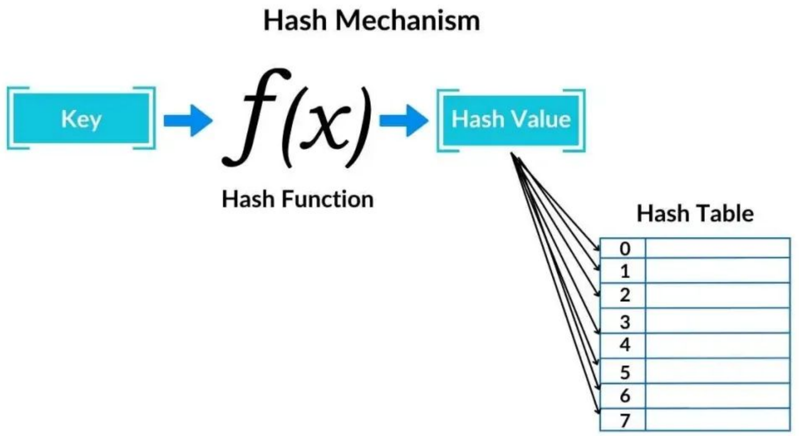
- new keys can be added with new values
- but all keys must be unique!

```
# add a new value to the dictionary
mlb_team_one['Kansas City'] = 'Royals'
```

```
# all keys must be unique
mlb_team_one['Kansas City'] = 'a different string'
```

Dictionary

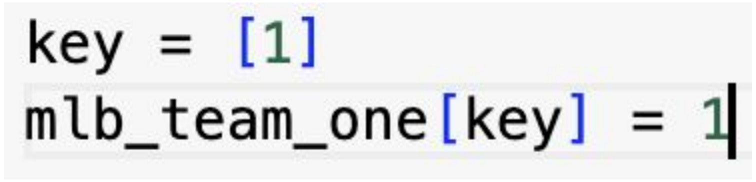
- Dictionaries efficiently search for the memory location of keys
- Special **hashing** functions are used
- Hashing is a process that uniformly maps many inputs to just a few inputs
- For example maybe 0-7 maps to 0-7
- AND 8-15 maps to 0-7
- AND 16-23 maps to 0-7
- Only immutable keys are hashable



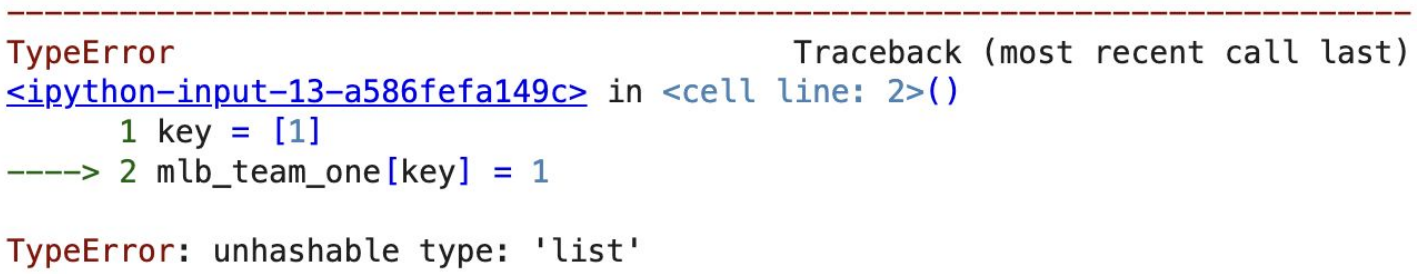
Dictionary

Critical Thinking

- can a list be a dict key?



```
key = [1]
mlb_team_one[key] = 1
```



```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-a586fefa149c> in <cell line: 2>()
      1 key = [1]
----> 2 mlb_team_one[key] = 1

TypeError: unhashable type: 'list'
```

Bespoke Dictionary

Bespoke Dictionary

- Make a dictionary by hand!

Bespoke Dictionary

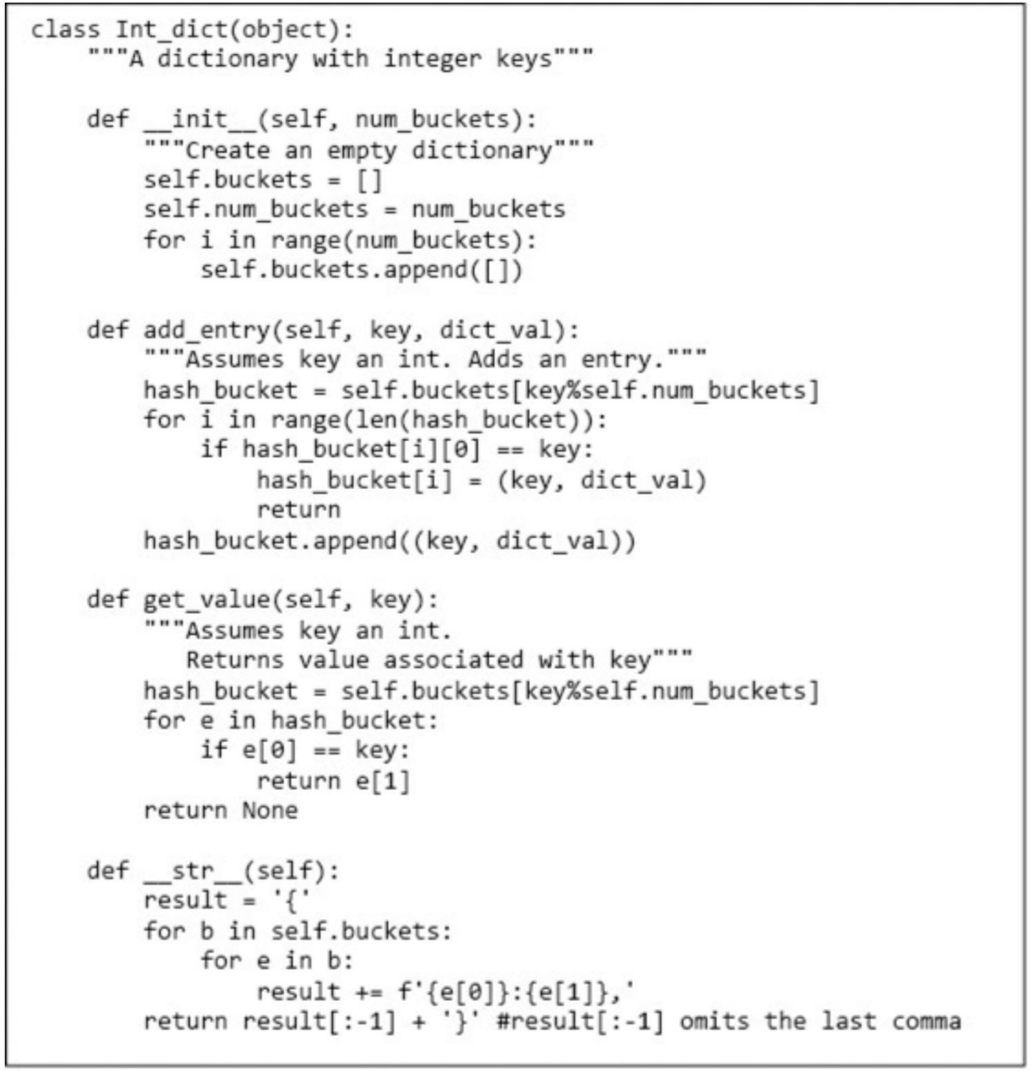
- Use a class to make a dictionary
- Conceptually, think of the dict as a matrix, i.e. a list of lists
- all the data will be stored inside the matrix as tuples of (key, value) pairs
- matrix row will be selected by hashing a key
- duplicate keys are not allowed

- class must have a method to add and entry, i.e. a key value pair
- class must have a method to get a value with a key
- class must have a method to hash the keys

- if more than one key should "hash" to the same row, append a new tuple to the row

Bespoke Dictionary

- Figure 12.7
- n.b. term hash_bucket may be used instead of row
- n.b. term buckets may be used instead of matrix



```
class Int_dict(object):
    """A dictionary with integer keys"""

    def __init__(self, num_buckets):
        """Create an empty dictionary"""
        self.buckets = []
        self.num_buckets = num_buckets
        for i in range(num_buckets):
            self.buckets.append([])

    def add_entry(self, key, dict_val):
        """Assumes key an int. Adds an entry."""
        hash_bucket = self.buckets[key%self.num_buckets]
        for i in range(len(hash_bucket)):
            if hash_bucket[i][0] == key:
                hash_bucket[i] = (key, dict_val)
                return
        hash_bucket.append((key, dict_val))

    def get_value(self, key):
        """Assumes key an int.
           Returns value associated with key"""
        hash_bucket = self.buckets[key%self.num_buckets]
        for e in hash_bucket:
            if e[0] == key:
                return e[1]
        return None

    def __str__(self):
        result = '{'
        for b in self.buckets:
            for e in b:
                result += f'{e[0]}:{e[1]},'
        return result[:-1] + '}' #result[:-1] omits the last comma
```

Explore Dictionary Code