Dictionaries

Brandon Krakowsky

Penn Engineering

1

Dictionaries

2

Dictionaries

- A *dictionary* (*dict*) is another way to store data, like a *list* or *set*, but as unordered key-value pairs. A *dictionary* is a set of *keys* and corresponding *values*.
  - Dictionaries are also known as *hashmaps* or *associative arrays* in other languages (e.g. Java)
- *Dictionaries* are extremely useful!
  - One use case is for storing several attributes (or data points) about a single thing
- To create a *dict*, use comma separated *key:value* pairs, in between curly braces {}
  - *keys* are simple data types (usually strings or ints)
  - *values* can be of any type
- Dictionaries are *mutable*, so once defined, elements can be changed

3

Dictionaries

- Here's a *dict* with some *keys* and associated *values* about a person

```
person = {'name': 'Zed', 'age': 39, 'height': 6 * 12 + 2}
print(type(person)) #A dictionary has a data type of dict
```

- We can get the value for a given key by using brackets []

```
print(person['name'])
```

- Or, we can use the built-in dict *get* method

```
print(person.get('name'))
```

- The *get* function is good to use, in case the *key* doesn't exist

```
print(person['state']) #KeyError will be generated if 'state' doesn't exist
print(person.get('state')) #this will return None (a null value) if 'state' doesn't exist
print(person.get('state', 'PA')) #this will return a default 'PA' if 'state' doesn't exist
```

Penn Engineering

4

Dictionaries

- Dictionaries are *mutable*, so elements can be updated or added

```
person['name'] = "John" #update value with key 'name'
person['age'] += 1 #increment value with key 'age'
person['college'] = True #add value with key 'college'
person['city'] = "Philadelphia" #add value with key 'city'
print(person)
```

- Check if a *key* exists in a dictionary

```
print('college' in person)
```

- Delete elements using the *del* keyword

```
del person['college']
print(person)
```

Penn Engineering

5

Dictionaries

- Dictionaries can include other dictionaries or lists as values

```
person['siblings'] = ['Cory']
person['siblings'].append('Betsy')
print(person)
```

- Or, we can add the key:value pairs from one dictionary to another using the built-in dictionary *update* method

```
person_attributes = {'marital status': 'married', 'children': 3}
person.update(person_attributes)
print(person)
```

Penn Engineering

6

Other Ways to Create a Dictionary

- You can initialize an empty dictionary
```
courses = {}
```
- Then add elements
```
courses['CIT590'] = 'Intro to Programming'
courses.update({'CIS545': 'Big Data'})
print(courses.get('CIS545'))
```
- Or create a dictionary from a list of tuples using the *dict* function
```
translation = dict([(1, 'uno'), (2, 'dos'), (3, 'tres')])
print(translation[1])
```

Penn Engineering

7

Other Ways to Create a Dictionary

- You can also create a dictionary from 2 separate lists
- Here's a list of *keys*
```
keys_lst = ['Joey', 'Fred', 'Katie']
```
- And a list of *values*
```
values_lst = [6, 4, '2 months']
```
- Make a sequence of tuples using the built-in *zip* function
```
zipped = zip(keys_lst, values_lst)
print(type(zipped)) #The type is zip
```
- Then create a dictionary from the *zip* using the *dict* function
```
kids_ages = dict(zipped)
print(kids_ages)
print(kids_ages['Fred'])
```

Penn Engineering

8