ECOR 1042
Data Management

Dictionaries

Recap Learning Outcomes Previous Lecture

- Review Python's `list` type
- More on lists: Slices, Aliasing, Functions that modify their list arguments, List methods, Nested lists
- Introduce Python's `tuple` and `set` types

References

- *Practical Programming*, 3rd ed.
  - Chapter 11, *Storing Data Using Other Collection Types*
    - *Storing Data Using Dictionaries* (pp. 214 - 222)
    - *Inverting a Dictionary* (pp. 222 - 223)
    - *Using the in Operator on Tuples, Sets and Dictionaries* (p. 223)
    - *Comparing Collections* (p. 224)

Lecture Objectives

- Introduce Python's `dict` (dictionary) type

Learning Outcomes (Vocabulary)

- Know the meaning of these words and phrases
  - Dictionary/map (type `dict`)
  - Key, value associated with a key, key/value pair, entry

Learning Outcomes

- Be able to evaluate expressions consisting of `dict` objects and some of the operations supported by that type
- Understand the key differences between lists, tuples, sets and dictionaries

Dictionaries

Dictionaries

- Python provides a built-in type named `dict`
- A `dict` is a collection of key/value pairs
- Like a `list` or a `set`, a `dict` is mutable
- As of Python 3.7, a dictionary is an ordered collection (the order in which the key/value pairs were inserted)

Some Notation - Caution

- Empty String: `s = ""`
- Empty List: `l = []`
- Empty Set: `s = set() //` `{}` gives a <u>Dictionary</u>
- Empty Tuple: `t = ()`

- 1-element String: `s = "1"`
- 1-element List: `l = [1]`
- 1-element Set: `s = {1}`
- 1-element Tuple: `t = (1,)`

Comma is known as the tuple constructor. Without the "," t will contain the <u>int 1</u>.

Creating `dict` Objects

- A `dict` object is created by an expression of the form `{`*`key1`*`: `*`value1`*`, `*`key2`*`: `*`value2`*`, ..., `*`keyN`*`: `*`valueN`*`}`
- Example: a directory containing course instructor's names and office numbers

```
>>> directory = {'Cristina': 'ME 4523',
        'Lynn': 'ME 4246'}
```

Creating `dict` Objects

- The *keys* are strings containing instructors' names
- Each key *maps* to a value (a string containing the instructor's office location)
- A key/value pair is also known as an *entry*
- Keys must be unique
- The same value can be associated with multiple keys; e.g., two instructors can share an office

Creating `dict` Objects

- Keys must be immutable objects; e.g., values of type `str`, `int`, `float`
  - Why?
- The values associated with keys do not have to be immutable

Creating `dict` Objects

- A dictionary remembers the order in which the entries (key/value pairs) were inserted

```
>>> directory = {'Cristina': 'ME 4523’,
      'Lynn': 'ME 4246', 'Safaa': 'ME 4476’,
      'Wafa': 'ME 4239', 'Rami': 'ME 4239’,
      'Don': 'ME 4522'}
>>> directory
{'Cristina': 'ME 4523', 'Lynn': 'ME 4246', 'Safaa':
'ME 4476', 'Wafa': 'ME 4239', 'Rami': 'ME 4239',
'Don': 'ME 4522'}
```

Creating `dict` Objects

- We can create an empty `dict` object and add the entries one-by-one

```
>>> directory = {}    # A dict, not a set!
>>> directory['Cristina'] = 'ME 4523'
>>> directory['Lynn'] = 'ME 4246'
>>> directory['Safaa'] = 'ME 4476'
>>> directory['Wafa'] = 'ME 4239'
>>> directory['Rami'] = 'ME 4239'
>>> directory['Don'] = 'ME 4522'
```

Operation: Associating a Value with a Key

- Syntax: $d[k] = v$
    - If key k is not in dictionary d, insert key/value pair k/v
    - If key k is in dictionary d, the old value associated with k is replaced by v

Operations: Retrieving Values

- Use the key to retrieve the value associated with a key

```
>>> directory['Cristina']
'ME 4523'
```

- Python raises a `KeyError` if the key is not present

```
>>> directory['Babak']
builtins.KeyError: 'Babak'
```

Operations: Retrieving Values

- The `get` method is similar to `d[key]`, except it returns `None` instead of raising a `KeyError` if the key is not in the dictionary

```
>>> directory.get('Cristina')
'ME 4523'
```

```
>>> directory.get('Babak')
>>> print(directory.get('Babak'))
None
```

Operations: Retrieving Values

- We can pass a second argument to `get`, which is the default value to return if the key is not in the dictionary

```
>>> directory.get('Cristina', 'unknown office')
'ME 4523'
```

```
>>> directory.get('Babak', 'unknown office')
'unknown office'
```

Operations: Updating a Key/Value Pair

```
# Lots of offices over the years...
>>> directory['Don'] = 'MC 3010'
# Change the value associated with 'Don'
>>> directory['Don'] = 'MC 3042'
>>> directory['Don'] = 'ME 4438'
>>> directory['Don'] = 'ME 4522'
```

- Each assignment replaces the old value associated with the key

```
>>> directory['Don']
'ME 4522'
```

Operations: `len()`

- `len()` returns the number of entries in a dictionary

```
>>> directory = {'Cristina': 'ME 4523',
        'Lynn': 'ME 4246', 'Safaa': 'ME 4476',
        'Wafa': 'ME 4239', 'Rami': 'ME 4239',
        'Don': 'ME 4522'}
>>> len(directory)
6
```

Operations: Checking Membership

- Use the `in` operator to check if a **key** is in a dictionary

```
>>> 'Safaa' in directory
True
>>> 'Babak' in directory
False
```

- Ca not use `in` to check if a value associated with a key is in a dictionary

```
>>> 'ME 4522' in directory
False
```

Operations: Remove an Entry

- Use the `del` operator to remove an entry

```
>>> del directory['Don']
>>> directory

{'Cristina': 'ME 4523','Lynn': 'ME 4246',
'Safaa': 'ME 4476', 'Wafa': 'ME 4239',
'Rami': 'ME 4239'}

>>> del directory['Babak']
builtins.KeyError: 'Babak'
```

Operations: Remove an Entry

- `pop` returns the value associated with a key, and removes the key/value pair (raises a `KeyError` if key is not present)

```
>>> directory = {'Cristina': 'ME 4523',
   'Lynn': 'ME 4246', 'Safaa': 'ME 4476',
   'Wafa': 'ME 4239', 'Rami': 'ME 4239’, 'Don': 'ME 4522'}
>>> directory.pop('Rami')
'ME 4239'
>>> directory
{'Cristina': 'ME 4523', 'Lynn': 'ME 4246', 'Safaa': 'ME 4476', 'Wafa': 'ME 4239', 'Don': 'ME 4522'}
```

Operations: Remove an Entry

- `pop` can return a default value if the key is not present

```
>>> directory = {'Cristina': 'ME 4523',
   'Lynn': 'ME 4246', 'Safaa': 'ME 4476',
   'Wafa': 'ME 4239', 'Rami': 'ME 4239', 'Don': 'ME 4522'}
>>> directory.pop('Rami', 'unknown office')
'ME 4239'
>>> directory.pop('Babak', 'unknown office')
'unknown office'
```

Operations: Looping Over a Dictionary

- A `for` loop iterates over all the **keys**

```
>>> directory = {'Cristina': 'ME 4523’,
    'Lynn': 'ME 4246', 'Safaa': 'ME 4476’,
    'Wafa': 'ME 4239', 'Rami': 'ME 4239', 'Don': 'ME 4522'}
>>> for name in directory:
...     print(name, directory[name])
...
Cristina ME 4523
Lynn ME 4246
Safaa ME 4476
Wafa ME 4239
Rami ME 4239
Don ME 4522
```

Note the order in which the keys are assigned to the variable `name`

Operations: Views

- Method `keys()` returns a *view* of all the keys in a dictionary

```
>>> names = directory.keys()
>>> for name in names:
...     print(name)
...
Cristina
Lynn
Safaa
Wafa
Rami
Don
```

Operations: Views

- Method `values()` returns a *view* of all the values in a dictionary

```
>>> offices = directory.values()
>>> for office in offices:
...     print(office)
...
ME 4523
ME 4246
ME 4476
ME 4239
ME 4239
ME 4522
```

Operations: Views

- Method `items()` returns a *view* of all the key/value pairs in a dictionary

```
>>> entries = directory.items()
>>> for entry in entries:
...     print(entry)
...
('Cristina', 'ME 4523')   # Each item is in a tuple
('Lynn', 'ME 4246')
('Safaa', 'ME 4476')
('Wafa', 'ME 4239')
('Rami', 'ME 4239')
('Don', 'ME 4522')
```

Carleton University

Common Dictionary Operations

- Review the textbook
- Know how to
  1. Get a key’s value
  2. Get an item from key's value
  3. Get all keys
  4. Get all values
  5. Get all items (i.e. key:value pairs)
  6. Update an item
  7. Update items
  8. Iterate over a dictionary

Dictionaries and Lists

Dictionaries and Lists

- Passing a dictionary to the `list()` function returns a `list` of all the keys in the dictionary, in insertion order

```
>>> list_of_keys = list(directory)
>>> list_of_keys
['Cristina', 'Lynn', 'Safaa', 'Wafa', 'Rami', 'Don']
```

Dictionaries and Lists

- Passing a view to the `list()` function returns a `list`

```
>>> list_of_keys = list(directory.keys())
>>> list_of_keys
['Cristina', 'Lynn', 'Safaa', 'Wafa', 'Rami', 'Don']
```

```
>>> list_of_values = list(directory.values())
>>> list_of_values
['ME 4523', 'ME 4246', 'ME 4476', 'ME 4239', 'ME 4239', 'ME 4522']
```

```
>>> list_of_items = list(directory.items())
>>> list_of_items
[('Cristina', 'ME 4523'), ('Lynn', 'ME 4246'), ('Safaa', 'ME
4476'), ('Wafa', 'ME 4239'), ('Rami', 'ME 4239'), ('Don', 'ME
4522')]
```

Can the values of a dictionary be…

- A tuple?
- A list?
- A set?
- Another dictionary?

Can the keys of a dictionary be…

- A tuple?
- A list?
- A set?
- Another dictionary?

Collections - Summary

	Strings	Lists	Sets	Tuples	Dictionary
Ordered	Yes	Yes	No	Yes	Yes*
Mutable	No	Yes	Yes	No	Yes
Duplicates allowed	Yes	Yes	No	Yes	No**
Notation	“ ”	[ ]	{ }	( )	{ : }
Declare an Empty	S = “”	L = [ ]	S = set()	T = ()	D = { }
Declare a 1-element	S = “1”	L = [1]	S = {1}	T = (1,)	D{1 : “hello”}

* As of Python version 3.7, dictionaries are *ordered*.
In Python 3.6 and earlier, dictionaries are *unordered*.
https://www.w3schools.com/python/python_dictionaries.asp

** No two items can have same key

Recap Learning Outcomes

- Know the meaning of these words and phrases
    - Dictionary/map (type `dict`)
    - Key, value associated with a key, key/value pair, entry
- Be able to evaluate expressions consisting of `dict` objects and some of the operations supported by that type
- Understand the key differences between lists, tuples, sets and dictionaries

ECOR 1042
Data Management

Dictionaries