CPSC 301: Computing in the Life Sciences
Lecture Notes 12:
Tuples, Sets and Dictionaries

2017-03-16

Jessica Dawson
jqdawson@cs.ubc.ca
http://www.cs.ubc.ca/~jqdawson

University of British Columbia
Department of Computer Science

2016 W2
Example using Sets

- Suppose we want to keep a list of the towns from which the UBC students come this year. When students registered we created a file `towns.txt` that contains the student id and the town for each student. We want to print out the towns without repeating them.

- A solution:

  ```python
  infile = open("towns.txt", "r")
towns = set()
for line in infile :
    words = line.split()
towns.add(words[1])

print "UBC Students come from the following towns:
for town in towns :
    print town
  ```
Summary of Facts about Sets and Frozensets

• Sets are used to store an unordered collection of distinct data
  – Each data item must be immutable
  – Supports common set operations: union, intersection, difference, element of (in), etc.
  – Data a and b are distinct if a is b evaluates to false

• A set is mutable
• A *frozenset* is immutable
• A frozenset is created by
  \[
  \text{frozenset(collection)}
  \]
  where collection is a list, a tuple, or another set
• A frozenset can be an element of another set or frozenset
• There is no shorthand notation for creating sets or frozensets
Dictionaries

• Suppose we want to extend the last problem to keep track of how many UBC students come from each town
  – we need to keep pairs (town, count) not single values
  – a set will not work in this case

• The right structure for that is a dictionary

• A dictionary is a collection of pairs of the form (key, value), i.e. they associate a value with a key
  – phone book: key ==> name, value ==> phone number
  – address book: key ==> name, value ==> email address

• Each key appears at most once (is unique) but a value can be associated with many keys

• Keys are immutable, but the values are not

• You can view a dictionary as a table with keys in column 1 and values in column 2
  – You may look up keys in column 1 and find out the value in column 2, but not vice-versa
Creating a Dictionary

• Usually we start with an empty dictionary:
  ```python
country_codes = {}
```

  or with some key/value pairs:
  ```python
country_codes = { 'Canada':'CAN', 'China':'CNN', 'Russia':'RUS'}
```

• Can add a new pair:
  ```python
country_codes['United States'] = 'USA'
```
  it associates the value 'USA' with the key 'United States' in `country_codes`

• Can change the value for a key:
  ```python
country_codes['China'] = 'CHN'
```

• Can retrieve an entry:
  ```python
country_codes['Canada'] will return 'CAN'
country_codes.get('Canada') will also return 'CAN'
```

• Can check if a key is in a dictionary:
  ```python
'Canada' in country_codes will return True
```

• Can remove an entry (key/value):
  ```python
def country_codes['United States'] removes the entry for United States
country_codes.pop('United States') removes entry and returns the value
```
Dictionary Methods

Dictionaries are objects and have their own methods. Their most important methods are:

• `dictionary.clear()`
  – empties the dictionary

• `dictionary.keys()`
  – returns the dictionary's keys; this object has no duplicates as keys are unique

• `dictionary.items()`
  – Returns the (key, value) pairs with all the items in the dictionary

• `dictionary.values()`
  – returns the dictionary's value; this object may contain duplicates.
  – ***If you want a list for any of the above three, you must also use the list() function

• `dictionary.update(other_dictionary)`
  – adds to dictionary the pairs from the other dictionary
Looping Over a Dictionary

- We can access all the items of a dictionary by looping over its keys:
  
  ```python
  for k in country_codes:
    print(k, ":", country_codes[k])
  
  or
  
  for k in country_codes.keys():
    print(k, ":", country_codes[k])
  
  We can access all the items of a dictionary by looping over its items:
  
  ```python
  for (k,v) in country_codes.items():
    print(k, ":", v)
  ```
Example

Here is a program that reads the file towns.txt and prints a list of the towns in alphabetical order and the number of students that come from each town:

```python
infile = open("towns.txt", "r")

# Create a dictionary with towns and number of students from them
towns = {}
for line in infile:
    words = line.split()
    town = words[1]
    if town in towns:
        towns[town] += 1
    else:
        towns[town] = 1

# Print towns and their count in alpha order
print "UBC Students come from the following towns:"
for town in sorted(towns):
    print town, ":", towns[town]

Note: The built-in function sorted() gets the dictionary keys and sorts them
```

Notes:
- Sets & Dictionaries
Summary of Facts About Dictionaries

• Any immutable data can be a key
  – For example: Booleans, any number type, strings, ...
• Any data can be a value
  – For example: Other dictionaries, files, numbers, strings, ...
• Dictionaries are mutable
  – The value associated with a key can change, but not the key
  – Keys can be added or deleted
• Many different mechanisms to access dictionary data
  – Get list of keys, list of values, loop over keys, test for key, ...
• Dictionary is not ordered
  – If you print a dictionary or iterate over a dictionary, Python will pick some arbitrary ordering (potentially different each time)
  – If you want the keys and/or values in a particular order, you need to create that ordering (eg: get the list of keys and sort it)
Summary

- Tuples are immutable lists
- Sets are mutable, unordered, unindexed collections of distinct heterogenous immutable data
- Frozensets are immutable sets
- Dictionaries are mutable, unordered collections of key-value pairs
  - The immutable, distinct keys serve as indexes
  - The values can be any data
- Sets and dictionary keys have very efficient tests for inclusion
- Ordering of sets and dictionary keys is not maintained
- Braces {} are used to create sets and dictionaries
- Brackets [] are used to access or modify values in a dictionary (the key is used as the index inside the brackets)
- Python has many functions, methods and operators which can create, examine and modify collections
Change Log

• Design Patterns for Sets: example name corrected to towns.py (from town_sets.py)
• Added a description for pop() to creating a dictionary slide
• Dictionary Methods: Clarified that .keys(), .values() and .items() do not return a list type directly
  – Python 2 returns a list, but Python 3 returns another type of object that you can iterate over which is similar to a list, but that can’t be indexed.