COMP10001 Foundations of Computing
Dictionaries and Sets

Semester 1, 2019
Tim Baldwin, Nic Geard, Farah Khan, and Marion Zalk

— version: 1520, date: April 4, 2019 —

Lecture Agenda

- Last lecture – Grok Worksheets 8–9
  - Iteration (cont.)
  - Lists
- This lecture – Grok Worksheet 11
  - Dictionaries
  - Sets

Reminders/Announcements

- Mid-semester test tomorrow; see the LMS for details
- Live(ish) Help now in full swing **every teaching day** (= Mon–Fri except for public holidays and the mid-semester break) for the remainder of semester, for the following times:
  - 1pm–2pm
  - 7pm–8pm
- Tim travelling next week, so you will have Nic as lecturer and Tim's email responsiveness will be severely hampered; for any email enquires (as always!) use:

  `comp10001-lecturers@lists.unimelb.edu.au`

Lecture Outline

So Many Numbers ...

- While computers like numbers, humans are not so numerically inclined. This can lead to mistakes in programming.
- Store the prices of drinks in a list: latte $3.50, chai $3.00, coke $2.00.

```
prices = (3.5, 3, 2)
```

- What's the cost of 478 lattes?

```
print(478 * prices[0]) # 0 is latte?
```

- Wouldn't it be loverly if we had a way of mapping directly from word (or some other attribute) to number?

Dictionaries

- As powerful as sequences are, we sometimes want to be able to store/access items relative to a (unique) key, in which case we use a "dictionary" (aka lookup table, map, hash, index, associative array, ...):

Phone list:	
Tim	41363
Nic	41301
Marion	41302
Farah	41303
Raph	41304

Favourite movies:	
My Neighbour Totoro	4.5 stars
Once Were Warriors	5 stars
The Big Lebowski	6 stars
Kung Pow	5 stars
The Holy Grail	5 stars

Dictionaries: Basic Operations

- Dictionary initialisation:

```
mydict = {}
mydict = {"latte": 3.50, "chai": 3.00}
```

- Adding items to a dictionary:

```
dictionary[KEY] = VALUE
```

- Accessing items in a dictionary:

```
mydict[KEY]
mydict.get(KEY)  # no KeyError
```

- Deleting items from a dictionary (in-place):

```
a = dictionary.pop(KEY)
```


Basic Dictionary Manipulation

```
>>> drinks = {"chai": 3.50, "latte": 3.75}
>>> 3*drinks["chai"] + 2*drinks["latte"]
18.0
>>> drinks["chai"] = 4.00
>>> drinks
{'chai': 4.0, 'latte': 3.75}
>>> drinks["capuccino"]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'capuccino'
>>> drinks.pop('latte')
3.75
>>> drinks
{'chai': 4.0}
```


More Dictionary Operations

- Key membership Test:

```
KEY in mydict # KEY found in mydict?
```

- Deleting the entire contents of a dictionary:

```
mydict.clear() # in-place
```

- Deleting an entry (no return value)

```
del mydict[KEY]
```


Removing Items with `del`

```
>>> drink_dict = {'chai': 3.50, 'latte': 3.75}
>>> drink_list = ['chai', 'latte']
>>> drink_tuple = ('chai', 'latte')

>>> del drink_dict['chai']; print(drink_dict)
{'latte': 3.75}

>>> del drink_list[0]; print(drink_list)
['latte']

>>> del drink_tuple[0]; print(drink_tuple)
TypeError: 'tuple' object doesn't support ...
```


Class Exercise

Write a function `word_freq(text)` that takes the string `text` as input, and returns a dictionary with the frequency of each word in `text`

Class Exercise

```
def word_freq(text):
    freq = {}
    for word in text.split():
        word = word.strip(".,:;!?'\"").lower()
        if word in freq:
            freq[word] += 1
        else:
            freq[word] = 1
    return freq
```

Dictionary Views

```
mydict.keys()   # returns a 'view' of keys
mydict.values() # returns view of values
mydict.items()  # returns (key,val) view
```

- a view will change if the underlying dictionary changes

```
>>> mydict = {'chai': 3.50, 'latte': 3.75}
>>> k = mydict.keys() ; print(k)
dict_keys(['chai', 'latte'])

>>> del drink_dict['chai']; print(k)
dict_keys(['latte'])
```


How to Use a Dictionary

- One common scenario for using a dictionary is to analyse trends in data, in instances where it isn't clear what values you are likely to observe in the data, e.g.:
  - common dog breeds/names
  - common names for a given breed
- Some handy functions:
  - `max()`: calculate the maximum value in a sequence
  - `sorted()`: generate a sorted list version of the argument (*without* mutating the argument)

Sets

- Sets are like dictionaries without keys:

```
>>> a = {1, 2, 3, 1, 2, 3, 1, 2, 3}
>>> print(a)
{1, 2, 3}

>>> 1 in a   # testing for membership
True

>>> myset = set('abracadabra')
>>> print(myset)
{'a', 'r', 'b', 'c', 'd'}

a = set() # gotcha: {} is the empty dictionary
```


Class Exercise

Write a function `max_word_freq(freqdict)` that takes a dictionary of word frequencies and returns the most frequent word

Lecture Outline

Sets

```
>>> a = set('abra')
>>> b = set('alac')
>>> a             # unique letters in a
{'r', 'a', 'b'}
>>> a - b         # set difference
{'b', 'r'}
>>> a | b         # set union
{'r', 'a', 'l', 'c', 'b'}
>>> a & b         # set intersection
{'a'}
```

Sets

- Like dictionaries, sets are not sequences (no order): you cannot slice or index them.
- Also like dictionaries, they are mutable with `add()`, `remove()`, `intersection_update()` (cf. `intersection` which *returns* the intersection set), `difference_update()` (cf. `difference` which *returns* the difference set), ...

Sets

```
>>> a = set('abra')
>>> a
{'r', 'a', 'b'}
>>> a.add('A') ; a
{'r', 'A', 'b', 'a'}
>>> a.add('A') ; a
{'r', 'A', 'b', 'a'}
>>> a.remove('a') ; a
{'r', 'A', 'b'}
>>> a.remove('a')
KeyError: 'a'
```


Sets

```
>>> a = {'apples'}
>>> b = {'oranges'}
>>> a.intersection(b)
set()
>>> a
{'apples'}
>>> a.intersection_update(b)
>>> a
set()
>>> b
{'oranges'}
```


Hashable

- Keys in dictionaries and elements in sets must be "hashable"
- All of Python's **immutable** built-in objects are hashable
  - `int`, `float`, `str`, `tuple`, `bool`
- Thus `list`, `dict`, `set` cannot be keys in dictionaries or elements in sets

Lecture Summary

- When and how are dictionaries commonly used?
- What is a dictionary, and what basic operations can we perform on a dictionary?
- What types can be keys in a dictionary?
- How do you create an empty dictionary?
- How do you add and delete from dictionaries?
- What is a set and how does it differ from a dictionary?