Science & Technology Facilities Council | UK Research and Innovation | NERC SCIENCE OF THE ENVIRONMENT

Python

Dictionaries

What is a *dictionary?*

A collection of key/value pairs

What is a *dictionary?*

A collection of key/value pairs

Keys are:

What is a *dictionary?*

A collection of key/value pairs

Keys are:

- Immutable

What is a *dictionary?*

A collection of key/value pairs

Keys are:

- Immutable

- Unique

What is a *dictionary?*

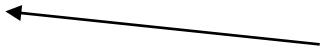
A collection of key/value pairs

Keys are:

- Immutable

- Unique

- Stored in order of entry



Since Python 3.7 – before were unordered

What is a *dictionary?*

A collection of key/value pairs

Keys are:

- Immutable

- Unique

- Stored in order of entry

No restrictions on values

What is a *dictionary?*

A collection of key/value pairs

Keys are:

- Immutable – they *cannot* be changed

- Unique

- Stored in order of entry

No restrictions on values

- Don't have to be immutable or unique

Create a dictionary by putting key:value pairs in { }

Create a dictionary by putting key:value pairs in { }

```
>>> birthdays = {'Newton' : 1642, 'Darwin' : 1809}
```

Create a dictionary by putting key:value pairs in {}

```
>>> birthdays = {'Newton' : 1642, 'Darwin' : 1809}
```

Retrieve values by putting key in []

Create a dictionary by putting key:value pairs in {}

```
>>> birthdays = {'Newton' : 1642, 'Darwin' : 1809}
```

Retrieve values by putting key in []

Just like indexing strings and lists

Create a dictionary by putting key:value pairs in {}

```
>>> birthdays = {'Newton' : 1642, 'Darwin' : 1809}
```

Retrieve values by putting key in []

Just like indexing strings and lists

```
>>> print(birthdays['Newton'])
1642
```

Create a dictionary by putting key:value pairs in {}

```
>>> birthdays = {'Newton' : 1642, 'Darwin' : 1809}
```

Retrieve values by putting key in []

Just like indexing strings and lists

```
>>> print(birthdays['Newton'])
1642
```

Just like using a phonebook or dictionary

Add another value by assigning to it

Add another value by assigning to it

```
>>> birthdays['Turing'] = 1612 # that's not right
```

Add another value by assigning to it

```
>>> birthdays['Turing'] = 1612 # that's not right
```

Overwrite value by assigning to it as well

Add another value by assigning to it

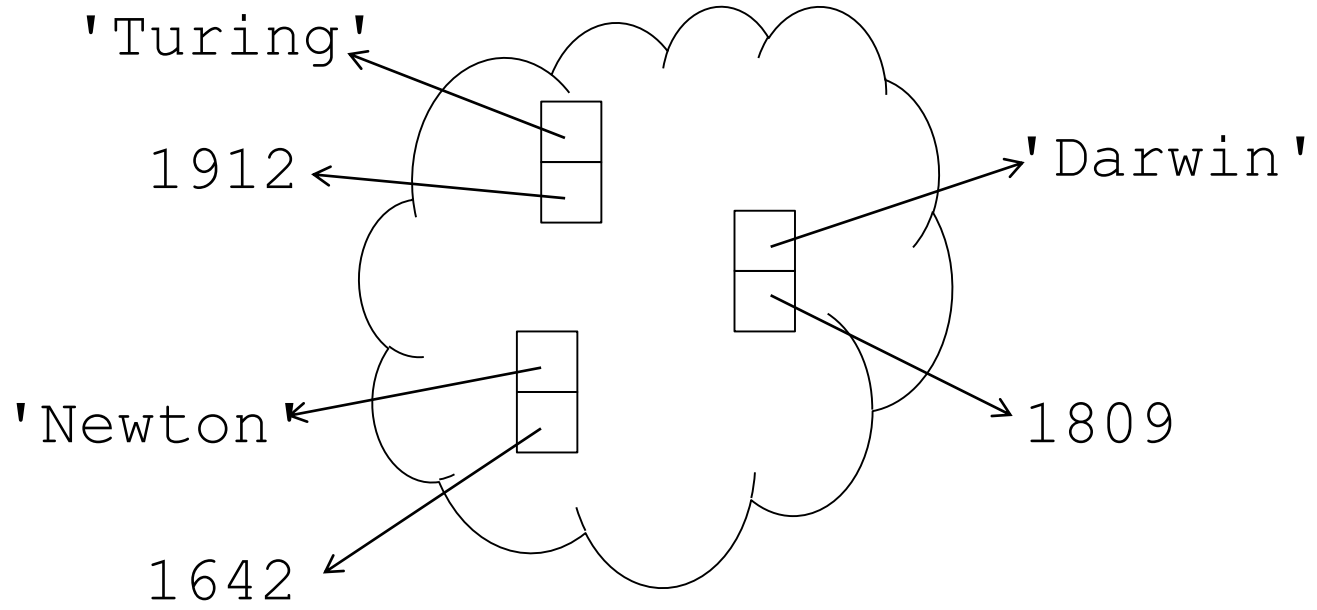
```
>>> birthdays['Turing'] = 1612 # that's not right
```

Overwrite value by assigning to it as well

```
>>> birthdays['Turing'] = 1912
>>> print(birthdays)
{'Turing' : 1912, 'Newton' : 1642, 'Darwin' : 1809}
```

Note: entries are *not* in any particular order

Note: entries are *not* in any particular order



Key must be in dictionary *before* use

Key must be in dictionary *before* use

```
>>> birthdays['Nightingale']
KeyError: 'Nightingale'
```

Key must be in dictionary *before* use

```
>>> birthdays['Nightingale']
KeyError: 'Nightingale'
```

Test whether key is present using `in`

Key must be in dictionary *before* use

```
>>> birthdays['Nightingale']
KeyError: 'Nightingale'
```

Test whether key is present using `in`

```
>>> 'Nightingale' in birthdays
False
>>> 'Darwin' in birthdays
True
```

Use `for` to loop over keys

Use `for` to loop over keys

Unlike lists, where `for` loops over values

Use `for` to loop over keys

Unlike lists, where `for` loops over values

```
>>> for name in birthdays:
...     print(name, birthdays[name])
```

```
Newton 1642
Darwin 1809
Turing 1912
```

Useful methods on dictionaries

`.keys(),.values(),.setdefault(<key>, <default>),.items()`

Useful methods on dictionaries

.keys(),.values(),.setdefault(<key>, <default>),.items()

```
>>> person = {"name": "Sarah", "height": 2}
>>> person.keys()
dict_keys(['name', 'height'])
>>> person.values()
dict_values(['Sarah', 2])
```

Useful methods on dictionaries

.keys(), .values(), .setdefault(<key>, <default>), .items()

```
>>> person = {"name": "Sarah", "height": 2}
>>> person.keys()
dict_keys(['name', 'height'])
>>> person.values()
dict_values(['Sarah', 2])

>>> person.setdefault('profession', 'Astrophysicist')
'Astrophysicist'
>>> person
{'name': 'Sarah', 'height': 2,
'profession': 'Astrophysicist'}
```

Useful methods on dictionaries:

`.items()` returns a list of tuples:

`[(<key>, <value>), (<key>, <value>)]`

```
>>> heights = {"Everest": 8848, "K2": 8611}
>>> heights.items()
dict_items([('Everest', 8848), ('K2', 8611)])

>>> for (mountain, height) in heights.items():
        print("{0} is {1}m high".format(mountain, height))

Everest is 8848m high
K2 is 8611m high
```