Session 3:

Strings and APIs

Andreas Bjerre-Nielsen

Agenda

1. Strings: manipulation, combination etc.
2. Containers - key based
3. Interacting with the web
4. Loading and saving files

Strings

Strings recap

What are strings? What do they consist of?

- Strings are sequences of characters
  - Characters can be whitespace.
- Python has two formats
  - `ascii` stings contain only English characters
  - `utf` strings contain also other European and Asian characters

String concatenation

How can I combine strings?

Strings can be added together:

In [2]:
```
s1 = 'police'
s2 = 'officer'
# s1 + s2
# s1 + ' ' + s2
```

String changing case

Can I alter the sentence-case of strings?

- Yes using the string methods `upper`, `lower`, `capitalize`. Example:

In [3]:
```
# s1.capitalize()
```

Substrings (1)

How can I check if a substring is contained in the string?

- in/not in

In [4]:
```
'pol' in s1
```

Out[4]:
```
True
```

Substrings (2)

How can I replace a specific substring?

- replace

In [5]:
```
s1.replace('po', 'ma')
```

Out[5]:
```
'malice'
```

Substrings (3)

Can I also access a string via indices? (in the sequence of characters)

- sequence form - slicing/indexing

In [7]:
```
s1[3:]
```

Out[7]:
```
'ice'
```

Strings quiz

Which Python object do strings remind you of?

- Lists work like strings.
  - Concatention (+, *) works the same way.
  - We check if element/character is contained with `in`.
  - We can slice and use indices for.

More about strings

There are many things about strings which we have not covered:

- There are many more string methods for splitting or combining strings etc.
- String formatting (http://www.python-course.eu/python3_formatted_output.php) is exceptionally useful when making a URL, printing etc.

Containers - key based

Containers recap

What are containers? Which have we seen?

-

Dictionaries (1)

How can we make a container which is accessed by arbitrary keys?

By using a dictionary, `dict`. Try executing the code below:

In [1]:
```
my_dict = {'Andreas':'Economist',
           'Snorre':'Sociologist',
           'Ulf':'Engineer'}
print(my_dict['Snorre'])
```

```
Sociologist
```

Dictionaries (2)

Dictionaries can also be constructed from two associated lists. These are tied together with the **zip** function. Try the following code:

In [2]:
```
keys = ['a','b']
values = [1,2]
key_value_pairs = zip(keys, values)

my_dict2 = dict(key_value_pairs)
print(my_dict2['b'])
```

```
2
```

Storing containers

Does there exist a file format for easy storage of containers?

Yes, the JSON file format.

- Is at the base a list or a dict.
- Looks like one:
  - `'{"a":1,"b":1}'`

Storing containers (2)

Why is JSON so useful?

- Extreme flexibility:
  - Can hold any list or dictionary of any depth which contains only float, int, str.
- Standard format that looks exactly like Python.

Interacting with the web

The web protocol

What is `http` and where is it used?

- `http` stands for Hyper Transm .. Protocol.
- `http` for transmitting the data when a webpage is visited:
  - the visiting client sends request for URL;
  - the server returns relevant data if active.

The web protocol (2)

Should we care about `http`?

- In this course we don't care explicitly about `http`.
- We use a Python module called `requests` as a `http` interface.
- However... Some useful advice - you should **always**:
  - use the encrypted version, `https`;
  - use authenticated connection, i.e. private login, whenever possible.

Markup language (1)

What is `html` and where is it used?

- `html` is a language for communicating how a webpage looks like and behaves.
  - That is, `html` contains: content, design, available actions.

Markup language (2)

Should we care about `html`?

- Yes, `html` is often where the interesting data can be found.
- Sometimes, we are lucky, and instead of `html` we get a JSON in return.
- Getting data from `html` will the topic of the subsequent scraping sessions.

APIs (1)

So when do we get lucky, i.e. when is `html` not important?

- When we get an Application Protocol Interface, i.e. `API`
- What does this mean?
  - We send an API query
  - We get an API response with data back in return, typically as JSON.

APIs (2)

So is data free? As in free lunch?

- Most commercial APIs require authentication and have limited free usage
  - e.g. Google Maps, various weather services
- If no authentication is required the API may be delimited.
  - This means only a certain number of requests can be handled per second or per hour from a given IP address.

APIs (3)

So how do make the URLs?

- An `API` query is a URL consisting of:
  - Server URL, e.g. `https://api.github.com`
  - Endpoint path, `/users/abjer/repos`
  - Query parameters,

APIs in Python (1)

How do make a simple query?

In [17]:
```
server_url = 'https://api.github.com/'
endpoint_path = 'users/abjer/repos'
url = server_url + endpoint_path
```

APIs in Python (2)

How can we send a query with the `requests` module?

In [29]:
```
import requests # import the module
resp = requests.get(url) # submit query with `get` and save response
```

APIs in Python (3)

How do extract something from the response?

In [28]:
```
# print(resp.text[:500])
```

APIs in Python (4)

Can we get something more meaningful or structured?

In [30]:
```
# resp_json = resp.json()
```

Loading and saving files

How to do input-output (IO) operations in Python

Text files

How can we save a string as a text file?

In [1]:
```
my_str = 'This is important.'

with open('my_file.txt', 'w') as f:
    f.write(my_str)
```

How can we load a string from a text file?

In [3]:
```
with open('my_file.txt', 'r') as f:
    my_str_load = f.read()
my_str == my_str_load
```

Out[3]:
```
True
```

JSON files

How can we save a JSON file?

The trick is to convert the JSON file to a string. This can be done with `dumps` in the module `json`:

In [33]:
```
import json

with open('my_file.json', 'w') as f:
    resp_json_str = json.dumps(resp.json())
    f.write(resp_json_str)
```

We can convert a string to JSON with `loads`.

File handling

How can we remove a file?

The module os can do a lot of file handling tasks:

In [34]:
```
import os
os.remove('my_file.json')
```