



Scripting Languages

Laszlo SZATHMARY

University of Debrecen
Faculty of Informatics

Lab #10

- downloading webpages
- JSON serialization

(last update: 2023-09-03 [yyyy-mm-dd])

2023-2024, 1st semester



Downloading webpages

```
4 import urllib.request
5
6 def main():
7     response = urllib.request.urlopen("http://python.org")
8     raw = response.read()
9     print(type(raw))    # bytes ←
10    # print(raw)
11
12    html = raw.decode("utf-8")
13    print(type(html))    # str ←
14    # print(html)
```

Since it's a common operation, let's put it in our `pylab.py` module in a function called `get_page()`. We want to use it like this:

```
3 from pylab import get_page
4
5 url = 'http://python.org'
6
7 def main():
8     print(get_page(url))
```

Downloading a URL (it can be an HTML, an image, etc.).
The downloaded object will appear in the filesystem:

```
4 import urllib.request
5
6 url = "http://www.nhdf1.org/uploads/NHB%20photos/HB0P2.jpg"
7
8 def main():
9     urllib.request.urlretrieve(url, "/tmp/forest.jpg")
```

Or, you can also use an external program to download something, e.g. `wget` :

```
3 import os
4
5 url = 'http://wallpapers.leovacity.be/images/Forrest_wallpaper.jpg'
6 to = '/tmp/forrest.jpg'
7
8 def main():
9     cmd = 'wget {url} -O {output}'.format(url=url, output=to)
10    print(cmd)
11    os.system(cmd)
```



calling an external program in another process

The **requests** module

The Python standard library has the following modules for downloading URLs:

- urllib (<https://docs.python.org/3/library/urllib.html>)
- urllib.request (<https://docs.python.org/3/library/urllib.request.html>)
- ...

However, if you want to do something more complex, they are not that easy to use :(

There is a very useful module called **requests** ([link](#)). It's not part of the standard library, you need to install it separately. But it's much simpler to use than the two modules mentioned above...

[Presentation from its author.](#)

Installation:

```
sudo apt-get install python3-pip  
sudo pip3 install pip -U  
sudo pip3 install requests -U
```

Run it once (if the package manager pip is not yet installed).

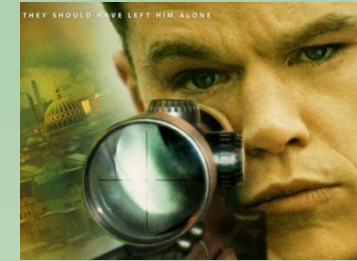
Update the pip package manager to the newest version.

install/update requests

How to use the **requests** module

```
>>> import requests
>>>
>>> url = "http://python.org"
>>> r = requests.get(url)
>>> print(r.text)
```

JSON serialization

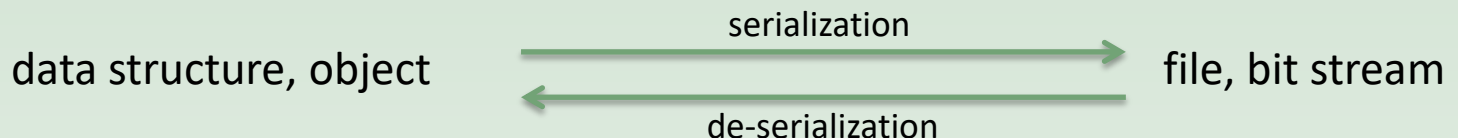


- <http://json.org/>
- <http://en.wikipedia.org/wiki/Json>

With the help of the **json** module you can serialize Python objects residing in the memory to JSON format.

Serialization

Converting a data structure or object to a format that can be stored in a file (or can be transmitted through network), and later either in the same, or in a different computing environment it can be „resurrected”.



Formats: XML, JSON, YAML, stb.



JSON can be often used as an alternative of XML

- also text-based, easily readable for humans
- also hierarchical
- can also be used for communication between applications
- simpler than XML
- not as verbose, shorter
- can be written / read faster

XML:

```
<Person>
  <FirstName>Homer</FirstName>
  <LastName>Simpson</LastName>
  <Relatives>
    <Relative>Grandpa</Relative>
    <Relative>Marge</Relative>
    <Relative>The Boy</Relative>
    <Relative>Lisa</Relative>
  </Relatives>
</Person>
```

JSON:

```
{
  "firstName": "Homer",
  "lastName": "Simpson",
  "relatives": [ "Grandpa", "Marge", "The Boy", "Lisa" ]
}
```


XML:

```
<persons>
  <person>
    <name>Ford Prefect</name>
    <gender>male</gender>
  </person>
  <person>
    <name>Arthur Dent</name>
    <gender>male</gender>
  </person>
  <person>
    <name>Tricia McMillan</name>
    <gender>female</gender>
  </person>
</persons>
```

JSON:

```
[
  {
    "name": "Ford Prefect",
    "gender": "male"
  },
  {
    "name": "Arthur Dent",
    "gender": "male"
  },
  {
    "name": "Tricia McMillan",
    "gender": "female"
  }
]
```

XML:

```
<settings>
  <path>/home/luke/Dropbox/Public</path>
  <user_id>123456</user_id>
  <auto_sync>True</auto_sync>
</settings>
```

(source: [Dropbox Publicus](#))

JSON:

```
{
  "path": "/home/luke/Dropbox/Public",
  "user_id": 123456,
  "auto_sync": true
}
```

JSON Syntax Rules

JSON syntax is a subset of the JavaScript object notation syntax.

- Data is in name/value pairs
- Data is separated by comma
- Curly brackets holds objects
- Square brackets holds arrays

JSON Values

JSON values can be:

- A number (integer or floating point)
- A string (in double quotes)
- A Boolean (true or false)
- An array (in square brackets)
- An object (in curly brackets)
- null

loading a JSON file

input file (person.json)

```
1 {
2     "last": "Doe",
3     "first": "John",
4     "age": 39,
5     "sex": "M",
6     "registered": true,
7     "salary": 70000
8 }
```

pretty print
(data structures are printed in a nicer way)

json module
(writing / reading json files)

```
3 from pprint import pprint
4 import json
5
6
7 def read_file():
8     f = open('person.json', 'r')
9     d = json.load(f)
10    f.close()
11
12    print(type(d))
13    pprint(d, indent=4)
```

 returned value: Python data structure
(here in the example: dictionary)

```
{
    'age': 39,
    'first': 'John',
    'last': 'Doe',
    'registered': True,
    'salary': 70000,
    'sex': 'M'}
```

output (dictionary)

loading a JSON string

```
7 def read_string():
8     s = """{
9         "path": "/home/luke/Dropbox/Public",
10        "user_id": 123456,
11        "auto_sync": true
12    }"""
13
14     d = json.loads(s)
15
16     print(d)
```

(see [settings.py](#))

loads
(s stands for string)

Summary

The *json* module takes a string (either from a file or from a normal string) and **builds a Python object** (e.g. list, dictionary). That is, we don't need to **parse the string manually**, the module performs this task for us!



Exercises

Asking my own IP address

Link: <https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=EnPy3.20120920g>

Downloading an image from Reddit

Link: <https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=EnPy3.20121126a>

Writing data structures to **file** in JSON format

(see [write_person.py](#))

```
3 import json
4
5 def write_file():
6     person = {
7         'last': 'Doe',
8         'first': 'John',
9         'age': 39,
10        'sex': 'M',
11        'registered': True,
12        'salary': 70000,
13    }
14
15    f = open('employee.json', 'w')
16    json.dump(person, f)
17    f.close()
```

Python dictionary

`json.dump(data_structure, file_handler)`

it creates a JSON string from the data structure and returns this JSON string

```
print(json.dumps(person))
```

pretty print

```
json.dump(person, f, indent=4)
```

pretty print,
ordered by keys

```
json.dump(person, f,
    indent=4, sort_keys=True)
```

Exercises



homework



1. [20120815k] prison break