PYTHON

Arthur Darcet
arthur@darcet.fr

MODULES - PACKAGES

Modules are text files with the **.py** extension

Packages are directories with a **__init__.py** file

IMPORTS

`import foo`

will import either the **foo.py** or the **foo/__init__.py** file

`from foo import bar` and `import foo.bar`

loads **foo/bar.py** or **foo/bar/__init__.py**

Lookup order is `sys.path`

RELATIVE IMPORTS

Inside a package:

```
from . import foo
from .foo import bar
from .. import foo
```

ALIASES

```
import numpy as np

from .foo import bar as foo_bar
```

THIRD PARTY PACKAGES

- Hosted on https://pypi.python.org
- Install with pip install pymongo
- Or with pip install -r requirements.txt

VIRTUALENV

```
python3 -m venv ~/.venvs/pton2

source ~/.venvs/pton2
```

RUN PACKAGES

```
python -m foo
```

will

- import foo/__init__.py
- run foo/__main__.py

FONCTION VARIADIQUE

```
def my_function(*args):
    print(args)
```

```
def my_function(**kwargs):
    print(kwargs)
```

```
args = (1, 'a')
kwargs = {'key': 'value'}
my_function(*args, **kwargs)
```

DECORATORS

```
@foo
def my_fn():
    return 3
```

```
@foo
class MyClass:
    attr = 3
```

exactly equivalent to:

```
my_fn = foo(my_fn)
MyClass = foo(MyClass)
```

MONGO DB

One database, multiple collections

One collection, multiple documents

A document is a dictionary (key-value mapping)

PYMONGO

Initialise the client:

```
client = pymongo.MongoClient('mongodb://…')
```

Get the "bar_collection" collection in the "foo_db" database:

```
col_object = client.foo_db.bar_collection
```

Insert an object:

```
col_object.insert({'whatever': ['some', 'values']})
```

List objects matching a query:

```
col_object.find({'tag': 'val'})
```

```
col_object.update_one({'_id': some_id}, {'$set': {'tag':
'val2'}})
col_object.delete_one({'_id': some_id})
```

HTTP SERVER

Create a server:

```
server = http.server.HTTPServer(('', 8000), Handler)
server.serve_forever()
```

`Handler` est une classe qui étend `http.server.BaseHTTPRequestHandler`

`do_GET()` est appelé par le serveur sur une instance de `Handler`

```
self.path # request path, starting with a leading /
self.headers # the request headers
self.send_response(200)
self.send_header('Content-Type', '…')
self.end_headers()
self.rfile.read() # returns the request body in bytes
self.wfile.write(b'…')
```