DAT python module

author: Malik Bougacha

FIXME, Lausanne

stdlib

External modules

- Very short presentation of modules I found great

- Very short presentation of modules I found great
- Because python can be very cute

stdlib

json

Start it

Do the job, in a nice manner

```
with open('myfile.json') as f:
    d = json.load(f)
```

```
with open('myfile.json') as f:
    d = json.dump(f)
```

Push it

```
class Dog:
    def __init__(self, name):
        self.name = name

    @classmethod
    def from_json(cls, d):
        new_object = cls()
        new_object.__dict__ = d

    @property
    def json(self):
        return json.dumps(self.__dict__)

doug = Dog("doug")
json_doug = doug.json
new_doug = json.loads(json_doug)
```

Of course, this is not safe for everyday use.

logging

Start it

simple logging system

```
logging.debug("debug message")
logging.info("debug message")
logging.warning("debug message")
logging.error("debug message")
```

You can also set the logging level very simply

```
root_logger = logging.getLogger()
root_logger.setLevel(logging.DEBUG)
```

Push it

- multiple logger
  - one per class
  - one per module
- multiple output
  - plain text
  - json python
  - yaml
  - Syslog
  - WHATEVERYOUMAYWANT

```
    handler = logging.handlers.SysLogHandler(address
= '/dev/log')
    root_logger = logging.getLogger()
    root_logger.addHandler()
```

threading/subprocessing

Start it

So I heard you like Mapping

```
def worker(number):
    #or anything
    return number**10

p = Pool(100)
p.map(range(3000), worker)
```

Work with multiprocessing (CPU bound) or threading (IO bound).

Push it

FIXME

External modules

requests

Start it

```
requests.get("http://google.ch")
```

Cookie and authentication for http. Act a bit like a browser

```
with request.Session() as session:
    session.get("http://google.ch")
```

Push it

Or twisted ?

sh

Start it

subprocess sucks

```
from sh import git
git.clone("https://github.com/rg3/youtube-dl.git")
```

with capture of stdout

```
from sh import ps
process = ps('-aux')
print(process)
```

Push it

Progressbar

Start it

imagine you have

```
for i in range(30000):
    longactios(i)
```

you wanna give that to someone but there is no way to know where the action is ? we need a bar

```
progress = ProgressBar()
for i in progress(range(30000)):
    longactios(i)
```

Push it

will use ‘len’ to calculate the percentage, you can specify it (if you use generator for example)

```
progress = ProgressBar()
for i in count(300):
    sleep(300)
```

You could even combine map with progressbar

even with a bouncing bar:

```
widgets = [FormatLabel('Bouncer: value %(value)d - '), Bou
pbar = ProgressBar(widgets=widgets)
for i in pbar((i for i in range(180))):
    time.sleep(0.05)
```

fabric

ssh on lamas steroid. you wanna do a uname :

```
def uname():
    run("uname -a")
```

then

```
fab uname -h boy
```

Blackmamba

Start it

- Write async socket connection
- Twisted is too long to read
- Yield the steps then:

```python
def request(host, port):
    yield blackmamba.connect(host, port)
    yield blackmamba.reset()
request("http://google.ch", 80)
```

Beautifulsoup

Start it

you don't want to learn to write

```
    /*[local-name() = 'root']/*[local-name() = 'elem']
```

- yes that's valid
- let's replace it with beautiful soup `python`
- much better

Push it

modify dom elements

SQLAlchemy

Start it

Imagine a one to one mapping between python object and line of sql table. Would be great right ? Let's do the connection first

```python
from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()
#and then
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String)
    fullname = Column(String)
    password = Column(String)

    def __init__(self, name, fullname, password):
        self.name = name
        self.fullname = fullname
        self.password = password
```

Push it

FIXME

Flask

- Light webframework

```
@route('/')
def index():
    return 'peewee'

@route('/<id>')
def gre(id)
    return id
```

combine with sql alchemy for a simple web app framework.

```
@route('/')
@with_db
def index(db):
    return db.select('all')
```

Push it

Doc is

Biography