Taming the Python

Alice Brenon
Hi!

- @abrenon
- guix user since ~February 2021
- mostly
  - functional programming (Haskell)
  - machine learning (Python)

Figure 1: A typical screenshot from #guix on libera.chat around 9 a.m. (Europe/Paris) on a weekday.
Use case
My PhD

- Natural Language Processing
- Digital Humanities
- Geographic Discourses in Encyclopedias

Advisors

- Denis VIGIER (ICAR)
- Frédérique LAFOREST (LIRIS)
- Ludovic MONCLA (LIRIS)

- Textometry
- Machine Learning
- Named-Entity Recognition
Managing corpora

Corpus
- l’Encyclopédie de Diderot & d’Alembert
- La Grande Encyclopédie
- Encyclopædia Universalis
- Wikipedia

Many states
- dependencies
- ongoing improvement

Figure 2: A (real) screenshot from a corpus folder
The Beast
The python’s biotope

Packages

- pip / requirements.txt
- conda
- virtualenv
- can pin dependencies to exact version!
- grown-ups use tox (controlled environments! reproducibility!)

Stateful

"The following minimal example will download and load default processors into a pipeline for English:

```python
c
import stanza
c
nlp = stanza.Pipeline('en')
c```

→ general sense of easiness
Notebooks: a love story...

- pure playground: world of "ideas"
- everything "just works"
- literate programming
- interactive! → explore data

Figure 3: Excerpt of a notebook generating figures


... gone horribly wrong

The (in)human shell

```python
# transformation de la liste en dataframe
df = pd.DataFrame(data, columns=['volume', 'numero', 'head', 'normClass', 'classEDdA', 'author'])
df = df.sort_values(['volume', 'numero']).reset_index(drop=True)
```

```python
df = pd.read_csv('../..../Data/EDdA-Classification/EDdA_dataframe_orginal.tsv', sep='	')
```

```python
df = pd.read_csv('../..../Data/EDdA-Classification/EDdA_dataframe_withContent.tsv', sep='	')
```

```python
df.dropna(subset=['content', 'contentWithoutClass', 'firstParagraph', 'ensemble_domaine_enccre'], inplace=True)
```

```python
#df = pd.read_csv('../..../Data/EDdA-Classification/EDdA_dataframe_withContent.tsv', sep='	')
```

```python
df = df.loc[(df['nb_words']>=15)]
```
... gone horribly wrong

**In [36]:**

```python
# Load the BERT tokenizer.
if model_chosen == "bert":
    print('Loading BERT tokenizer...')
    tokenizer = BertTokenizer.from_pretrained(tokeniser_bert)
elif model_chosen == "camembert":
    print('Loading CamemBERT tokenizer...')
    tokenizer = CamembertTokenizer.from_pretrained(tokeniser_bert)
```

**Out [36]:**

```
Loading CamemBERT tokenizer...
```

**Out [36]:**

```
Downloading: 0% | 0.00/811k [00:00<, ?B/s]
```

**Figure 4: Dynamically downloading a part of the logic**
Install packages

!pip install transformers==4.10.3
!pip install sentencepiece

Collecting transformers==4.10.3
  Downloading transformers-4.10.3-py3-none-any.whl (2.8 MB)
    |████████████████████████████████| 2.8 MB 5.2 MB/s

Collecting sacremoses
  Downloading sacremoses-0.0.47-py2.py3-none-any.whl (895 kB)
    |████████████████████████████████| 895 kB 47.4 MB/s

Collecting tokenizers
  Downloading tokenizers-0.10.3-cp37-cp37m-manylinux_2_5_x86_64.whl (3.3 MB 49.6 MB/s)

Collecting regex
  Downloading regex==2019.12.17

Collecting numpy
  Downloading numpy-1.17

Figure 5: Calling shell constructs
... gone horribly wrong

```
!pip install git+https://github.com/ClaudeCoulombe/FrenchLefffLemmatizer.git &> /dev/null
!pip install spacy
!python -m spacy download fr_core_news_sm
```

Figure 6: Executing remote code
... gone horribly wrong

```python
embedding_dim = 300

embeddings_index = {}

# f = codecs.open('/Users/lmoncla/Documents/Data/Models/cc.fr.300.vec', encoding='utf-8')
# f = codecs.open('/Users/dm2l/Documents/cc.fr.300.vec', encoding='utf-8')

f = codecs.open('/home/alice/Logiciel/FastText/cc.fr.300.vec', encoding='utf-8')
```

Figure 7: The problem of data location
... gone horribly wrong

Setup colab environment

```python
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

Install packages

```bash
# !pip install zeugma
# !pip install plot_model
```

Figure 8: An attempt to solve it by centralizing the data
Guix to the rescue !
Containers

- isolate `$HOME`  
- control the environment  
- restrict access to the network

`guix shell --container --manifest manifest.scm`

In Python-only world:

```
python3 -m venv test
source test/bin/activate
pip install -r requirements.txt
```

`requirements.txt ⇝ manifest.scm`

`but python ∈ manifest.scm`!
Containers

- isolate `${HOME}`
- control the environment
- restrict access to the network

```
guix shell --container \
    -m manifest.scm
```

**in python-only world**

```
pip install -r requirements.txt
```

```python
source test/bin/activate
```

```
requirements.txt \→ manifest.scm
```

Happy birthday Guix! – Guix to the rescue!

2022-09-16
Containers

- isolate \${HOME}
- control the environment
- restrict access to the network

```bash
guix shell --container
    -m manifest.scm
```

**in python-only world**

```bash
python3 -m venv test
source test/bin/activate
pip install -r requirements.txt
```

requirements.txt ⇝ manifest.scm

**but**

python ∈ manifest.scm!
Let the errors guide you

- Access static data?

        guix shell -C -m manifest.scm --expose=$PATH_TO_DATA

Happy birthday Guix! – Guix to the rescue!
Let the errors guide you

- Access static data?
  
guix shell -C -m manifest.scm --expose=$PATH_TO_DATA

- Access the network?
  
guix shell -C --network -m manifest.scm
Let the errors guide you

- Access static data?
  ```
guix shell -C -m manifest.scm --expose=$PATH_TO_DATA
  ```

- Access the network?
  ```
guix shell -C --network -m manifest.scm
  ```

*mind the real reason behind the error!*
Mixing `guix` and `pip`

`guix import ⇒ pip ⊂ guix`: not always true

- non-free license?
- too ugly to package?
- no source available?

**Container bonus**

Use `pip` inside a container!

`guix shell -C -N -m nice-packages.scm`

`pip3 install dgl`

Happy birthday Guix! – Guix to the rescue!
Example: the FIDLE Mooc

- (https://fidle.cnrs.fr/)
- Deep Learning
- Set of notebooks
- Static Datasets
- Setup instructions (~requirements)

(use-modules (gnu packages machine-learning)
(gnu packages python-science)
(gnu packages python-xyz))

(define fidle-datasets
(load "fidle-datasets.scm"))

(packages->manifest (list fidle-datasets
jupyter
python-h5py
python-matplotlib
python-pandas
python-pytorch
python-pyyaml
python-scikit-image
python-scikit-learn
tensorflow))

Figure 9: A manifest to follow the FIDLE Mooc

guix shell -C -N -m manifest.scm \ 
--expose="/path/to/the/notebooks" -- jupyter-notebook
What about static data?

- huge (binary) models: `fasttext`, `stanza`
- input data
- they can be packaged too!
- replace arbitrary locations with symbolic ones: environment variables

```python
# ---- datasets location
#
datasets_dir = os.getenv('FIDLE_DATASETS_DIR', False)
if datasets_dir is False:
    error_datasets_not_found()
# Resolve tilde...
datasets_dir=os.path.expanduser(datasets_dir)
```

Figure 10: Some code in FIDLE interfacing the datasets
FIDLE datasets

```
(package
...
(source (origin
  (method url-fetch/tarbomb)
  (uri "https://fidle.cnrs.fr/fidle-datasets.tar")
  (sha256
    (base32
      "04qb7vdwsqxilc3w9j6303hj87gqz1my0pspljh346p1ji974k1v")))))
(build-system copy-build-system)
...
(native-search-paths
  (list (search-path-specification
    (variable "FIDLE_DATASETS_DIR")
    (files '("fidle-datasets")))))
...)
```

Figure 11: Using search-path-specification to package the datasets

Happy birthday Guix! – Guix to the rescue!
Thank you!

https://gitlab.liris.cnrs.fr/abrenon/fidle-mooc