Useful resources for Python

Aritz Bercher

May 26, 2020

Abstract

This will be a diary containing a list of reference concerning python programming, what I discovered, remarks, and everything I could reuse later.

Contents

1 Useful resources

In this section I will try to list some useful resources (mainly websites) to look for information concerning python.

- The official documentation: `https://docs.python.org/3.6/tutorial/index.html`
- `http://www.pythonforbeginners.com/basics`
- A website quite beginner friendly: `https://www.python-course.eu/python3_course.php`
- paperswithcode.com present implementation of some AI papers: `https://paperswithcode.com/`

2 Good coding practice

- When opening or saving a file, it is good to specify the encoding.
- When accessing the value of a key which may not exist of a dictionary, the `get` method is the right way:

```
entities = message.get("entities", [])
```

3 Journal

05.10.17 ~ Starting with python, anaconda and jupyter notebook
Since a couple of days I try to assimilate the bases of Python.
I did a little latex file called "Python doc and how to start" where I explain in detail how to start with python.
At first I was a bit puzzled by the fact that Python needs an interpreter and not a compiler (the difference is quite well explained in the book of Charles Severance p.8). I'm still a bit confused by the notions of Python virtual environment and in which way it differs from a virtual machine. I found a nice explanation there:
`https://blog.docker.com/2016/03/containers-are-not-vms/`
but it seems to be a reading a bit too advanced for my current knowledge. I started by installing Anaconda and it seems to be one of the easiest way to begin using Python.
I was a bit confused at the beginning before I had the following understanding of programming. First we write a script in a text file (generally inside a IDE) and then use a compiler (by pressing on a button in the IDE for instance) which translates it into some machine language and gives it to the CPU. And because of this I was failing to see how it could be done in several ways and what could be a virtual environment. Now I see it a bit differently. I see Python as a bunch of files (which include all the packages that we add little by little) describing a function which is called when we use `python arguments` in the terminal. So having a virtual environment is just telling to the computer "now when I say `python` in the terminal, I want you to use these files there".
It seems there is no standard editor for python script. I've been recommended to use jupyter notebook, pycharm and sublime. At first jupyter notebook didn't behave how I wanted. When I was in a virtual environment with python 2 installed and launched jupyter notebook from the command line, I was only able to create python 3 kernel. The problem was that I had not installed the package jupyter notebook in this virtual environment so it was going back to the root (the python installed "normally" on my linux) to find it. But then I installed the package jupyter notebook in all my virtual environment and it solved the problem.
I learned a bit about **list**, **tuples** and **dictionaries** (globally data structures) here:
`https://docs.python.org/3.6/tutorial/datastructures.html#dictionaries`
and abount sequence type here:
`https://docs.python.org/3.6/library/stdtypes.html?highlight=list#sequence-types-list-tup`
and a bit about **class** and **methods** here:

http://networkstatic.net/python-tutorial-classes-objects-methods-init-and-simple-example
I also discovered the keyword `pass` which is explained here:
https://stackoverflow.com/questions/13886168/how-to-use-the-pass-statement-in-python
I also had difficulty to change the language of the dictionary of texmaker but I found the solution here (although it is supposed to be for windows):
http://www.swisswuff.ch/wordpress/?p=166

06.10.17 ∼ yield, iterables, generators, with
I learned a bit about the keyword `yield`, **iterables** and **generators** here:
https://stackoverflow.com/questions/231767/what-does-the-yield-keyword-do
As I'm trying to create a python script to download songs from internet and that `youtube_dl` seems to be the right command to this in python, I tried to read this page:
https://github.com/rg3/youtube-dl/blob/master/README.md#embedding-youtube-dl
But for this I needed to understand what the keyword `with` was. I found several links:
https://stackoverflow.com/questions/1369526/what-is-the-python-keyword-with-used-for#11783672
https://stackoverflow.com/questions/3012488/what-is-the-python-with-statement-designed-f
effbot.org/zone/python-with-statement.htm
https://docs.python.org/release/2.5.2/lib/typecontextmanager.html https://www.python.org/dev/peps/pep-0343/
But these pages require already a good basic knowledge of python so I didn't get everything.
Edit2 (20.02.18): I read again this page about the `with` statement. It is a kind of safety command. If we open a file, and then do some manipulation with it, we want the file (or maybe the flow) to be closed if an error occurs. This command `with` guarantees it.
Edit: I found this page concerning **generators** which is quite beginner-friendly:
https://wiki.python.org/moin/Generators

08.10.17 ∼ Adding a password to jupyter notebook
I would like to get rid of this page appearing each time I start a jupyter notebook where I have to insert the password. I found this page which might tell me how to do it:
https://jupyter-notebook.readthedocs.io/en/stable/public_server.html
but as I didn't find this jupyter folder, I wanted to use the command `find` but and as I would like to filter out all the "permission denied" errors, I read this page:
https://unix.stackexchange.com/questions/42841/how-to-skip-permission-denied-errors-when-42842
I wasn't sure what was the symbol | for but I found it p.14 of the bashguide pdf given by TheAlternative.
I also found out what a symbolic link is there:
https://kb.iu.edu/d/abbe
As I didn't find the config file I created one as indicated and I received this message

from the console indicating where the configuration file for jupyter was put:

```
Writing default config to: /home/aritz/.jupyter/jupyter_notebook_config.py
```

Even though I created this passord when I was inside the virtual environment my_python35 it seems that now when I enter the command `jupyter notebook` in this environment or on the normal one I'm directed to a page where I need to enter the password I chose (and no longer copy paste the code appearing in the terminal or clicking on the link appearing in the terminal).

11.10.17 ~ youtube dl for python (Roman)
Roman sent me a link to a one of his projects on GitHub where he used python and youtube_dl:
`http://nbviewer.jupyter.org/github/rlyapin/video_summarization/blob/master/video_summarization_with_dl_embeddings_%28part_II%29.ipynb`.

12.10.17 ~ Vectorized operations in Python: lists, Creating and manipulating matrices with NumPy, Comprehension lists, asterisk (how to transform a list into positional arguments for a function), zip, and other tools
I started looking at the exercises of the course CIL. Everything which follows comes more or less from the first exercise sheet of CIL. I learned a few command to use NumPy objects and methods with the exercise sheet 1. The library NumPy seems to provide the basic tool to do array manipulation and vectorized operations which are computationally lighter for python that using for loops. If I understood well what is written here:
stack overflow: difference pandas and numpy
NumPy provides the basis and pandas some more advanced tools.
Comprehension lists seems the tool to do a lot of vectorized operations, or to apply a function iteratively to entry of one or more lists. It is presented here:
list comprehensions
I learned about the `zip` function. Somehow, it inverses the dimensions in a list.
`https://stackoverflow.com/questions/13704860/zip-lists-in-python#13704903`
The exercise 1 instruction sheet contains a list of basic vectorized operations and the associated commands in python.
I learned useful informations about the use of asterisk in order to take a list as input, and expand it into actual positional arguments in the function call, or deal with extra argument in a function, here:
`https://stackoverflow.com/questions/5239856/foggy-on-asterisk-in-python`
and there:
`https://stackoverflow.com/questions/400739/what-does-asterisk-mean-in-python`
The last exercise was nice. It consisted in computing for some points in $\mathbb{R}^2$ their likelihood depending on two different Gaussian model assumption and assigning each point to the most likely model.

14.10.17 ~ More arrays with NumPy
I found this nice tutorial to use NumPy:
NumPy Tutorial
and learned how to manipulate these arrays.

15.10.17 ~ Beginning of first ML project
I think I should decide a naming convention for my files and directories. I think I will always try to put Capital letters at the beginning of the name of a folder and lower case for files.
There is a nice explanation of what this SSH encryption is here:
Understandin the SSH encryption and connection process
I also started the first project of the course Machine Learning of ETH.

16.10.17 ~ ML project, trying to get started: Sumatra, Scikitlearn
I kept trying to do this ML project but there are plenty of things to get familiar with before being able to start like Sumatra and Scikitlearn (c.f. journal project ML1 for more details). I came across this page which gives a very good introduction to machine learning and how to use python for it:
basic tutorial on scikit learn

17.10.17 ~ Python coding guideline, Python modules and packages, python classes and metaclasses, doctstrings, API, signatures, magic methods
I found this general guideline for python coding style:
`https://www.python.org/dev/peps/pep-0008/`
I learned a few things concerning python packages and modules in these two webpages:
importing subpackages
difference module and packages
For more informations about modules and packages this page looks very detailed:
`https://docs.python.org/2/tutorial/modules.html`
Concerning the Scikit-learn part of the project we are encouraged to read the following webpages:
coding guidlines
APIs of scikit-learn objects
rolling your own estimator
I think I start seeing why we use this scikit-learn. This story of fit, fit transform etc... is described in the "API's of scikit-learn objects" section. If I get it right, the organizers of the project want us all to use the same kind of syntax with the function/method/API that we use in our code.
I found another good website for python documentation: `http://www.pythonforbeginners.com/basics`
And in particular for docstrings:

docstrings
I will put all good tutorials for python in the latex file "how to use python".
This two webpages explain in a complementary way what a python signature is (more or less):
difference between function declaration and signature
how to read the signature of a function
There's a beginning of information concerning API's here:
`https://wiki.python.org/moin/API`
I found an interesting page about classes and magic methods here:
magic methods and operator overloading
I also found this great webpage about objects, classes, and metaclasses in python:
`https://stackoverflow.com/questions/100003/what-is-a-metaclass-in-python#6581949`

18.10.17 ~ More on scikit-learn
The organizers published a very helpful sequence of slide which I called

`final_submission_instructions`

which explains a bit better how to use this scikit-learn framework.

19.10.17 ~ Trying to follow the instructions provided for the ML project, git, pipeline
Otherwise I finished following the instructions given by in the file called "final submission instructions". I really didn't do much as everything was implemented but I got a bit an idea of what these different tools are. There are still plenty of things I need to figure out (see my ml project journal for more details).
Concerning scikit-learn pipeline, I found these two pages which seem very good:
Stack Overflow: sklearn pipeline
and
`http://scikit-learn.org/stable/auto_examples/model_selection/grid_search_text_feature_extraction.html`

24.10.17 ~ Flake8, Submitting my code for ML project 1
I will try to submit my code (even though it is exactly what the tutors gave us as a model). I ran the command `flake8` and corrected what was pointed out. It seems to indicate style errors (like additional spaces at the end of a line) in order to obtain a code which follows some standards. Here is a webpage which introduces it well:
Flake8

26.10.17 ~ Scikit-Learn tutorial
I started to read the first scikit-learn tutorial:
`http://scikit-learn.org/stable/tutorial/basic/tutorial.html`

28.10.17 ~ Saving objects with Pickle, getting familiar with scikit-learn, and merging numpy arrays

I learned a few things about the `pickle` tool in these tow pages:
`https://stackoverflow.com/questions/8968884/python-serialization-why-pickle#8968969`
`https://www.thoughtco.com/using-pickle-to-save-objects-2813661`
I progressed in the reading of the scikit-learn tutorial. I learned fitting, predicting, splitting the data set into a train set and a test set, saving python object with `pickle`, reshaping a data set, the K-nearest neighbor algorithm, from these pages:
`http://scikit-learn.org/stable/tutorial/basic/tutorial.html`
`http://scikit-learn.org/stable/tutorial/statistical_inference/settings.html`
`http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html`
I also learned how to merge/concatenate numpy arrays with the commands `np.r` and `np.c` here mainly:
`https://docs.scipy.org/doc/numpy/reference/generated/numpy.r_.html`
and here
`https://docs.scipy.org/doc/numpy/reference/generated/numpy.c_.html`

30.10.17 ~ Plots for support machine algorithm with Python

In the scikit-learn documentation there is this nice script to output plot to delimit the boundaries of the areas computed by the support machine algorithm:
`http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html#sphx-glr-auto-examples-s`
I read an interesting page about randomness in python here:
`https://stackoverflow.com/questions/7029993/differences-between-numpy-random-and-random-`
It looks like numpy.random is the most convenient library.

31.10.17 ~ Sequence type objects, mutable and immutable types, built in functions, lists, cross-validation

`https://www.tutorialspoint.com/python/list_pop.htm` I learned plenty of interesting things about lists and more generally sequence type objects here:
Official documentation on list tuple and range
Among which the way memory is shared when doing copy of an object (which Roman told me about). This is also related to:
How to create a multidimensional list
Concerning the topic of memory allocation, I think this topic is actually the "mutable and immutable types" topic in python. I read something very interesting about it here:
`https://stackoverflow.com/questions/8056130/immutable-vs-mutable-types`
it reminds me of pointers in C++. I have the impression that immutable types variable behave like "normal" variable while mutable looks like pointers.
Here is a list of built-in functions, i.e. functions that we can use in a python script without having to use `import`:

https://docs.python.org/3.6/library/functions.html
There is also a very nice list of questions and answers for python programming here: Programming FAQ
I learned a few things on for loops and iterable here:
https://wiki.python.org/moin/ForLoop
I learned what the

%s

symbol means here:
https://stackoverflow.com/questions/997797/what-does-s-mean-in-python#997807
I learned how to do cross-validation in python here:
http://scikit-learn.org/stable/tutorial/statistical_inference/model_selection.html It presents several ways to do cross-validation:

1. The naive implementation, where we cut manually the data set and use a normal for loop.
2. An implementation using "Cross validation generators" like `KFold` which produce distinct set of indices to separate between training and testing set, and using a regular for loop.
3. An implementation using "Cross validation generators" like `KFold` which produce distinct set of indices to separate between training and testing set, and using a comprehension list instead of a regular for loop.
4. A method using the `cross_val_score` helper, which is a kind of wrapper for the previous method.

Now all these methods, tell us how to compute the average score across the k different folds for one value of the parameter of our estimation model that we want to choose. To choose it well one might use a for loop over a grid of values for this parameter using one of the method above. Or one can use a wrapper called `GridSearchCV` like displayed below (example taken from the tutorial):

```
lasso = Lasso(random_state=0)
alphas = np.logspace(-4, -0.5, 30)

tuned_parameters = [{'alpha': alphas}]
n_folds = 3

clf = GridSearchCV(lasso, tuned_parameters, cv=n_folds, refit=False)
clf.fit(X, y)
scores = clf.cv_results_['mean_test_score']
scores_std = clf.cv_results_['std_test_score']
```

01.11.17 ∼ cross-validation and grid searching for parameters of ML algorithm, ploting with python

I kept reading the tutorial presented above concerning cross validation. It presented a tool called `GridSearchCV` designed to for an optimal parameter among a set of parameters for a regression/classification algo using cross validation.
Also, if I understood right, stratified cross validation is used if we have a classifier, and then the observations are distributed in the different folds in a way assuring that there are (more or less) the same number of observations with a given label in each class.
I learned what the `enumerate` command does here:
`https://stackoverflow.com/questions/22171558/what-does-enumerate-mean`
I found this nice tutorial for ploting with python:
`http://matplotlib.org/users/pyplot_tutorial.html`
There is a more detailed page in the scikit-learn tutorial dedicated to cross-validation but I didn't read it yet:
`http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation`
To create a list of values between two values with a given increment the tool is numpy `arange`
I tried to do the exercise at the end of this page:
`http://scikit-learn.org/stable/tutorial/statistical_inference/model_selection.html`
but I obtain different results with different methods and it is not coherent with the solution. I need to investigate furthermore. Actually the main difference between what I had done and the solution was that I was shuffling the observations in the data and not the solution. The solution is also restricting itself to only the first 150 observations (I don't know why), which also had an influence. The file which contains my solution was called "scikit-learn ex4" with an underscore.
For the normalization of data, I also found this scikit-learn page:
sk-learn: normalization
It is part of a bigger tutorial on how to preprocess the data. The tool for normalization seems to be `StandardScaler` which is presented here:
sklearn.preprocessing.StandardScaler

04.11.17 ∼ PCA with python, Exceptions and error handling

I looked a bit at this PCA, and here is what I learned. If we have some centered data $X \in \mathbb{R}^{n\times p}$ (meaning the average value of each column of X is 0), the covariance matrix of the p features is estimated by

$$C = \frac{1}{n-1} X^T X.$$

We aim to somehow reduce the dimensionality of X while keeping the most information, i.e. the most variance. We perform Singular value decomposition (c.f. Stack Exchange: PCA and SVD, Stack Exchange: Loadings and Eigenvectors for instance) on X and we get

$$X = USV^T$$

where $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{p\times p}$ and S is a diagonal matrix, and

$$U^T U = \mathrm{Id}, \qquad V^T V = \mathrm{Id}$$

which implies that

$$C = VLV^T \qquad \text{with} \qquad L = \tfrac{1}{n-1}S^2.$$

This last decomposition is called Principal Component Analysis (PCA). The columns of V are eigenvectors of C, also called **principal axis** or **principal directions**. The columns v_i of XV are called **principal components**.
If I understood it right, when we do in python

```
pca = PCA(n_components=20).fit(X)
```

we keep only the first 20 columns of V (get $\tilde{V} \in \mathbb{R}^{p\times 20}$) and of U (get $\tilde{U} \in \mathbb{R}^{n\times 20}$), and only the first 20 columns of L (get $\tilde{L} \in \mathbb{R}^{20\times p}$). So we have a new data matrix $\tilde{X} \in \mathbb{R}^{n\times p}$ which satisfies

$$\tilde{X} = \tilde{U}\tilde{L}\tilde{V},$$

but has only rank 20. I think that the attribute

```
pca.components_
```

contains $\tilde{X}\tilde{V}$, but Actually I'm quite confused. I tried to follow an example of wikipedia to see precisely what is what, but it turns out that the pca method of sk-learn doesn't give me the expected result.
From what I read online, if the sole purpose that we have is to compute the SVD, then this might be better:
scipy.linalg.svd

At the end I couln't figure out what this .components_ attribute was. I wrote a jupyter notebook called PCA_SVD_test where I tried to find out but I failed. Ultimately I posted a question on Cross Validated:
Cross Validated: meaning of pca.components_
Edit: Someone answered. It seems that PCA assumes centered columns and in my example the columns were not centered...

Edit (05.05.18): Someone posted a more detailed answer on my stack exchange/cross validated tread, with a project used as illustration.
I read something interesting about **exceptions and errors handling** here:
Stack Overflow: exception handling
If I understand well exceptions in python are objects of some predefined classes. What I don't understand is how to trow an error of a specific type. For instance if I want to check that the arguments received by a function are of the appropriate type and throw an error if they aren't how do I do?
Edit (29.10.19): See entry of 29.11.18.

08.11.17 ~ Scikit-learn developer guide
To see if an estimator satisfies the sk-learn standards, I found this tool on the sk-learn website:

```
>>> from sklearn.utils.estimator_checks import check_estimator
>>> from sklearn.svm import LinearSVC
>>> check_estimator(LinearSVC)  # passes
```

18.11.17 ~ Preprocessing data and more on Cross Validation
I found this blog for preprocessing. It gives some sample code with scikit-learn:
blog about feature selection and preprocessing
For Cross Validation with scikit learn, this page contains a lot of informations:
sk-learn: cross validation

25.11.17 ~ Image processing and view as block
To reshape an array into array of arrays (blocks) there is a command called `view_as_blocks`.
The way it works is explained there:
view_as_blocks
The library is

30.11.17 ~ Overview of what is possible with sklearn
I found this link toward a page which contains plenty of useful examples for some machine learning tasks performed with sklearn tools: sklearn: examples and in particular something about "Feature union with heterogeneous data sources":
Feature union with heterogeneous data sources

14.12.17 ~ Indexing, Contiguous arrays
I stumbled upon something strange in my project about ECG where I wanted to transform some array of complex values into something in $\mathbb{R}^2$ (an array with twice more columns) and received an array with twice more rows. I posted the question on stack overflow:
Stack overflow: subsetting affects view
and someone said that I should look at the page about indexing in python:
`https://docs.scipy.org/doc/numpy-1.13.0/reference/arrays.indexing.html#detailed-notes`
Someone also indicated this nice glossary for numpy:
scipy: numpy glossary
Some answer to my question lead me to the topic of "Contiguous", "C-contiguous", and "Fortran-contiguous", which is quite well explained here:
Stack overflow: difference contiguous and non-contiguous arrays.

16.12.17 ~ Oversampling unbalanced data sets
Someone on Piazza indicated this webpage for oversampling minority classes in case of

unbalanced data set: imblearn.over_sampling.SMOTE

17.12.17 ~ jupyter notebook doesn't find a package inside an environment

Yesterday, I tried to use some function from imblearn inside a jupyter notebook that I had started from my environment ml_project. I received the error message

```
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1-d51ab97bec0a> in <module>()
----> 1 from imblearn.over_sampling import SMOTE, ADASYN

ModuleNotFoundError: No module named 'imblearn'
```

even though imblearn was install in the conda environment ml_project. As a student explained it to me on piazza, the problem was that jupyter notebook wasn't itself installed in the environment. The solution can be found here:
stack overflow: packages from conda env not found in jupyter notebook

01.01.18 ~ Literals variables, f-strings

I learned about **literals variables** in python here:
Stack overflow: What are literals in python
I read about a new kind of string literal called "**f-strings**" here:
docs.python.org: what's new in python 3.6

08.01.18 ~ autoreload, magic methods

I learned about `autoreload` here:
ipython.org: autoreload
and a bit about magic IPython commands here:
`https://scientificallysound.org/2017/06/01/ipython-magic-commands/`
"IPython magic commands provide extra functionality to develop code, and some are especially useful when writing code in Jupyter notebooks."

11.01.18 ~ Useful tips for Jupyter notebook, Decorators, del statement, Oriented Object Programming

I found some nice tips for Jupyter notebooks here:
`https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/`
I learned about decorators which look like `@something` here:
`https://en.wikipedia.org/wiki/Python_syntax_and_semantics#Decorators`
I learned about the `del` statement here:
`https://docs.python.org/2/tutorial/datastructures.html#the-del-statement`
I read some of this very nice introduction of Oriented Object Programming in python here:

python-course: python3 object oriented programming
The concept of encapsulation is interesting. Everything related to the manipulation of attributes of the given class should be implemented as a method.

12.01.18 ~ Property, Getters, Setters
I learned a bit about the keyword `@property` here:
python-course: python3 properties
From what I understood, it is only a way to avoid using explicitly getters and setters, and more precisely, being able to update some class where the attributes were previously public and had no getters and setters, without having to change the script based on the original implementation of the class.
I also found this page which provides more details (but less OOP background):
programiz: property
I also read this about the reason why one would use "ambiguous" assignment:
stackoverflow: When and how to use the builtin function property in python

13.01.18 ~ Updating output of a cell, fetching images from the web
Talking with Roman, I discover a nice trick: if one wants to run a loop and having the ouput of the cell updated for each loop (for a plot for instance), one can use this:
`https://stackoverflow.com/questions/24816237/ipython-notebook-clear-cell-output-in-code#24818304`
I tried to create a new data set to reproduce the experiment of the first lecture. I tried to use the code presented here:
`https://github.com/hardikvasa/google-images-download/blob/master/google-images-download.py`
but the results I got were not very satisfying. I also lacked a way to put it in the right subfolders, with the right naming convention. The guys from the slack channel told me that they did this last part manually.
Maybe I could use this script for this last goal:
`http://forums.fast.ai/t/dogs-vs-cats-lessons-learned-share-your-experiences/1656/37`
Someone on the forum, gave a link to this code:
`https://github.com/ohmeow/google_image_downloader`
I found also this website which seems to be good source of images:
`http://www.image-net.org/`
and this one:
`http://yfcc100m.appspot.com/`
In this page they give some bash script to put a percentage of images in one folder in a different subfolders:
`http://forums.fast.ai/t/bash-scripts-for-creating-sample-datasets/2235`
and also this one:
stackoverflow: linux bash move x percentage of files from each folder

14.01.18 ~ Classes and subclasses, classmethod and staticmethod
I found this page which gives a good introduction to classes and **inheritance** in python:
`http://www.jesshamrick.com/2011/05/18/an-introduction-to-classes-and-inheritance-in-pyth`
I learned about `@classmethod` and `@staticmethod` here:
Stackoverflow: classmethod and staticmethod for beginner.
Edit (19.10.18): I learned read this page which gives some further indications concerning the above mentioned decorators:
What is the difference between @staticmethod and @classmethod
As explained there:
What does the classmethod do in this code?
the classmethod decorator is useful to declare **inheritable alternative constructors**.

18.01.18 ~ next, iter
I learned a bit about the `next` and `iter` functions there:
`https://docs.python.org/2/library/functions.html#iter`

22.01.18 ~ Debugging
Jeremy Howard (fastai) advised to read this concerning debugging in python:
`https://pythonconquerstheuniverse.wordpress.com/2009/09/10/debugging-in-python/`

23.01.18 ~ Displaying progress bar with tqdm

I discovered the command `tqdm`, looking at the code of the method `TTA` of the class `learner`

09.02.18 ~ Reinstalling Python and Anaconda
Since I had to reinstall Kubuntu, I need to reinstall Pyhton and Anaconda. Python 3.5 seems to have been installed by default:

```
aritz@aritz-ThinkPad-T460p:~$ python -V
Python 2.7.12
aritz@aritz-ThinkPad-T460p:~$ python3 -V
Python 3.5.2
```

Then I installed anaconda after having downloaded the infos from the website of Anaconda and followed these indications.
To know if I was able to reinstall an environment from the folder (inside

`/media/aritz/My Passport/Sauvegardes/Kubuntu/Home_29.01.18/anaconda3/`

) I copied one of these folders present on my hard drive (my_anaconda35) into the envs folder of my new anaconda folder (the one created on my home by the installer in

`/home/aritz/anaconda3/envs/`

). To see if it had worked I tried to activate and it looks like it worked:

```
aritz@aritz-ThinkPad-T460p:~$ conda env list
# conda environments:
#
my_python35              /home/aritz/anaconda3/envs/my_python35
root                  *  /home/aritz/anaconda3

aritz@aritz-ThinkPad-T460p:~$ source activate my_python35
(my_python35) aritz@aritz-ThinkPad-T460p:~$ jupyther notebook
jupyther: command not found
(my_python35) aritz@aritz-ThinkPad-T460p:~$ anaconda-navigator
(my_python35) aritz@aritz-ThinkPad-T460p:~$
```

I guess that it works. Rather copying each one of my environments, I will try to download use again the .yml file from fastai to rebuild the environment. If later (for my ML projects for instance I need to rebuild an environment) I will copy it from my backup of home of the 29.01.18 on my passport.
I created a tread here:
`https://groups.google.com/a/continuum.io/forum/#!topic/anaconda/XZN2YqTTBBA`
to know if it is possible to reinstall the environment the way I did it.

11.02.18 ~ isinstance, duck typing
I learned a bit about `isinstance` and *basestring* here:
stackoverflow: difference between type and isinstance
It touches the topic of "**duck typing**" which seems to be a widespread concept in python. If I understood it well, it aims at developping tools which treat variables depending on their "superficial behaviour" (*if it looks like a duck, swim like a duck...*) rather than depending on their specific implementations.

13.02.18 ~ pandas (manipulating tables), seaborn (data visualization)
I read this introduction to the pandas library:
`https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f`
It seems to be the right tool to manipulate excell-like sheets (typically .csv) files.
I discovered the existence of seaborn, a library to do data visualization like histograms.

20.02.18 ~ glob for globbing in python, checking that CUDA is working and that we are using GPU, opening and reading/writing files within Python
I discovered the function `glob` which is used to do globing (like in bash) when manipulating file and directory names:
python.org: glob

The notebook 1 of fastai has been updated and now contains two commands to verify that we are using the GPU:

`torch.cuda.is_available()`

and

`torch.backends.cudnn.enabled`

Both give me True, so I think that I'm using my GPU correctly (even though it is very slow).
I read a bit about **double and simple quotes in pyhton** here:
stackoverflow: single quotes vs double quotes in python
I learned how to **open text files inside python** and how the `open` and `close` commands have to be combined with the `with` command here:
`https://pythontips.com/2014/01/15/the-open-function-explained/`
It seems to be also working for .csv files (since a .csv file is nothing but a text file, where the columns are separated by comas and the rows by a jump of line). One can also open other types of files as explained here :
`http://www.blog.pythonlibrary.org/2010/09/04/python-101-how-to-open-a-file-or-program/`
This last link contains plenty of useful links at the end for

- **os module** documentation
- **subprocess module** documentation
- Reading and Writing a File
- Working With File Objects

22.02.18 ∼ Pandas (again), merging data frames

Since I need to manipulate **data stored in tables** for the Rossmann data set, I tried to learn a bit more about the library pandas. I first read this page:
`https://pythonforengineers.com/introduction-to-pandas/`
but it was a bit too elementary.
This tutorial was a bit better:
`https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f`
and I completed it with some stack overflow pages:
`https://stackoverflow.com/questions/14734533/how-to-access-pandas-groupby-dataframe-by-k`
`17302673`
`https://stackoverflow.com/questions/39755981/explain-how-pandas-dataframe-join-works#`
`39756074`
Then I looked again at the third notebook of fastai, and there is a nice example of merging of several tables using different methods of pandas. This page helped me understand how to **merge data frames**:
`https://chrisalbon.com/python/data_wrangling/pandas_join_merge_dataframe/`

I learned a bit about **regular expressions** and the command `re.sub()` here:
`https://docs.python.org/2/library/re.html`
and here:
`https://docs.python.org/3.5/library/re.html#re.sub`

24.02.18 ~ Trying to solve ordinary differential equations with python
To help Joffrey with his master thesis I tried to solve find a way to solve numerically a non linear, ordinary equation of the second degree with python. It seems that PyDSTool is a good tool for that:
`http://www2.gsu.edu/~matrhc/PyDSTool.htm`
but from its information guideline it seem to require to use Python 2.7. I created a new environment called `mypy27` and then I tried to install numpy version 1.4.1 with conda but it failed. So I tried to use pip and it looked like it worked:

```
(mypy27) aritz@aritz-ThinkPad-T460p:~$ pip install numpy==1.4.1
Collecting numpy==1.4.1
  Downloading numpy-1.4.1.tar.gz (2.2MB)
    100% || 2.2MB 426kB/s
Building wheels for collected packages: numpy
  Running setup.py bdist_wheel for numpy ... done
  Stored in directory: /home/aritz/.cache/pip/wheels/b9/cc/6a/e92386a68ce853f545c7122f5a
Successfully built numpy
Installing collected packages: numpy
Successfully installed numpy-1.4.1
```

But then when I tried to install the version of matplotlib which was installed I ran into trouble (it looks like it wasn't available anymore) so I installed the most recent version. Buth then it turned out that this version of matplotlib required a newer version of numpy. So I upgraded numpy (by uninstalling it with pip and reinstalling it). Here is a summary of the version of the three packages required to install PyDSTool:

```
(mypy27) aritz@aritz-ThinkPad-T460p:~$ python
Python 2.7.14 |Anaconda, Inc.| (default, Dec  7 2017, 17:05:42)
[GCC 7.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> import scipy
>>> import matplotlib
>>> print numpy.__version__
1.7.1
>>> print scipy.__version__
0.9.0
>>> print matplotlib.__version__
2.1.2
```

Then there is an issue with the fact that my system has is 64 bit system and not 32 but if I'm not mistaken it is only for something optional, so I will hope that it works well. If not, I don't really know what I will have to do.
Then, I'm supposed to install and unzip a file in a directory of my choice, and update my system's path so that Python can find the package, plus several other complicated things. Since Joffrey seems to be able to solve his problem with matlab, I don't think I will try further to install this because I'm a bit worried I might hurt my computer and screw up the whole system.
There seem to be alternatives like sympy:
stackoverflow: how do I solve a non linear equation in sympy?
or scipy:
`https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html`
and
`https://stackoverflow.com/questions/40832798/solve-ordinary-differential-equations-using`
and (this last one seems very good):
stackoverflow: numerical ODE solving in python
and
scicomp stackexchange: how to get ODE solution at specific time points

26.02.18 ~ Pandas DataFrame and Series, Numpy Datetimes and Timedeltas
I learned a bit about the main types of the `pandas` library called `DataFrame` and `Series`:
`https://pandas.pydata.org/pandas-docs/version/0.15.2/dsintro.html`
If I get it right, one can see a DataFrame object as a dictionary of Series object where the rows are aligned depending of their index (each Series object has one index).
I also learned about numpy `datetime64` and `timedelta64`, which seem to be a way of encoding dates and times and manipulating (`timedelta64` allows to look at differences between times).

05.03.17 ~ pandas grouby, pandas rolling
I learned a bit about the command `groupby` there:
`https://www.tutorialspoint.com/python_pandas/python_pandas_groupby.htm`
It allows to perform operations (filtering, applying functions, transforming) on entries (rows) of the table depending of the value they have for one (or several) column(s). It allows to somehow modify the structure of the data frame: we can create from a 2D data frame a kind of 3D data frame, where the index of the new dimension corresponds to the value of one of the columns of the original data frame.
I learned about `pandas.DataFram.rolling` there:
`https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.rolling.html`

06.03.18 ~ pandas MultiIndex, pandas .loc
I learned a bit about **multi-level indexing with pandas** there:
`https://pandas.pydata.org/pandas-docs/stable/advanced.html`
I also learned a bit about pandas `.loc` method to subset parts of a DataFrame there:
Stackoverflow: loc function in pandas
It seems to be a safety measure useful when one wants to subset rows of an array which satisfy some conditions relative to the values of certain columns.
Edit (05.05.18): I found this very nice page explaining how to do **indexing on Pandas Series and DataFrame** with `.log` and `.iloc` and what is the difference between the two:
`https://stackoverflow.com/questions/31593201/pandas-iloc-vs-ix-vs-loc-explanation-how-ar`
`31593712`

23.03.18 ~ Trying to install pytorch, __iter__ and __next__
I tried to reproduce in a Jupyter notebook the N-gram language model example from this page:
pytorch.org: word embeddings tutorial
but with my personal conda environment `mypy36`. But pytorch wasn't installed so I installed it. But even after that I still got the same error message in the jupyter notebook. I found this page:
`https://github.com/pytorch/pytorch/issues/4827`
and I tried

```
conda update conda
```

but I ran into error. Some googling made me realize that it was maybe because I didn't have the right privileges in the conda environment I was running and indeed, after deactivating it, I could run the command. Then I ran

```
conda install mkl=2018
```

but it is still not working. I'm still getting

```
No module named 'torch'
```

when I run a cell in Jupyter notebook containing

```
import torch
```

but if I enter

```
python -c "import torch"
echo $?
```

in the terminal I obtain 0 which indicates that torch is installed in mypy36 (according to this page).
After reading the message of soumith here:

`https://github.com/pytorch/pytorch/issues/909`
I suspect that it is due to the fact that maybe jupyter notebook is not installed directly in the environment mypy36. So I entered

```
conda install jupyter
```

and started again a jupyter notebook and this time the command worked!
An important remark is that the library is called `torch` and not `pytorch` inside python (meaning one has to use `import torch`).

I learned a bit about the `__iter__()` and `__next__()`: if I understood it well, these methods are only some "internal machinery", and are not really supposed to be used by the user. They are used if we want to iterate over an object with a for loop. For instance if one has an instance MyInstance of a class MyClass where these two methods are implemented (and follow the conditions explained in the doc) one can use

```
for i in MyInstance
```

30.03.18 ~ Initializing list of lists (strange behaviour)
I stumbled on this problem while trying to initialize a list of list of zeros:
`https://en.wikibooks.org/wiki/Python_Programming/Lists#List_creation_shortcuts`

05.04.18 ~ Find python version, package version
In order to know **which version of python** we are using (typically inside a conda environment), a simple way is to type

```
python -V
```

In order to list all the modules installed within an environment (with both `pip` and `conda`) together with their version, use

```
conda env export -n myenv
```

(note that `conda list` will only list the packages installed with conda).

10.04.18 ~ from future, lambda
I learned about the command

```
from __future__ import whatever
```

here:
Stackoverflow: What is future in python used for and how/when to use it?
I learned about the **lambda expressions** and more precisely the `lambda` keyword here:
`https://docs.python.org/3/tutorial/controlflow.html`

12.04.18 ~ iterable object, iter(), next()
Looking at the object `trainloader` appearing in this tutorial:
`http://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`
I realized the following thing: an object (here `trainloader`) might have the `__iter__()` method implemented (as can be seen in its source code here) but it doesn't necessarily mean that we can use

```
trainloader.next()
```

to obtain manually the items it contains one by one (typically when we want to test some procedure on only one item instead of doing it on the whole data set). To be able to do it one has to use first the function *iter*() like this:

```
dataiter = iter(trainloader)
images, labels = dataiter.next()
```

03.05.18 ~ reversed, assert for sanity check
I learend about `reversed` here:
`https://docs.python.org/3.5/library/functions.html#reversed` and saw it in application in this tutorial:
`https://pytorch.org/tutorials/beginner/nlp/advanced_tutorial.html`
More precisely the code was:

```
for bptrs_t in reversed(backpointers):
    best_tag_id = bptrs_t[best_tag_id]
    best_path.append(best_tag_id)
```

Also in this tutorial, I ran across `assert` which is used for sanity check, or more precisely to verify that something holds (if it doesn't it produces an error). The code was:

```
assert start == self.tag_to_ix[START_TAG]
```

05.05.18 ~ Python string formatter: old style, new style, f-format
I found this page which explains pretty well how strings can be formatted (often used when **printing values of variables**):
`https://pyformat.info/`
An example of application of the new style found in this tutorial is

```
print('Image name: {}'.format(img_name))
```

This is completed by the more recent **f-strings** formats:
`https://docs.python.org/3/whatsnew/3.6.html#whatsnew36-pep498`

06.05.18 ~ Installing spaCy, pip vs pip3, spaCy vs NLTK

I started **installing Spacy**, following this webpage:
https://www.analyticsvidhya.com/blog/2017/04/natural-language-processing-made-easy-using
e2%80%8bin-python/
After having activated mypy36, I entered

```
sudo pip3 install spacy
```

But it didn't work properly. spacy was installed on the root environment instead of mypy36. I think that it is because I don't pip3 in mypy36, but only pip. That made me wonder what was the **difference between pip and pip3**. If I understood well this answer I think that pip is supposed to be used (at least when working directly on the root) for python 2.7 whereas pip3 is supposed to be used for python 3.5 and onward. I guess that when we activate a virtual environment (like mypy36) using python 3, then we can use pip or pip3 interchangeably (as long as both pip and pip3 are installed in the given virtual environment). Actually it seems that in this last case, it is really the same thing if I believe what is written in this post:
https://stackoverflow.com/questions/40189744/should-i-use-pip-or-pip3-to-install-python3
and also the output of this command:

```
(mypy36) aritz@aritz-ThinkPad-T460p:~$ pip -V
pip 10.0.1 from /home/aritz/anaconda3/envs/mypy36/lib/python3.6/site-packages/pip (python
```

I uninstall spacy from the root environment with

```
sudo pip3 uninstall spacy
```

and tried spacy in mypy36 with

```
sudo pip install spacy
```

but it seems to have failed:

```
(mypy36) aritz@aritz-ThinkPad-T460p:~$ spacy -V
/home/aritz/anaconda3/envs/mypy36/bin/python: No module named spacy
```

So I tried to install it again but using the -H flag (following the advice from some warning but I got this message telling me that it was already installed:

```
(mypy36) aritz@aritz-ThinkPad-T460p:~$ sudo -H pip install spacy
[sudo] password for aritz:
Requirement already satisfied (use --upgrade to upgrade): spacy in /usr/local/lib/python2
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.7 in /usr/lib/python2
Requirement already satisfied (use --upgrade to upgrade): murmurhash<0.29,>=0.28 in /usr/
Requirement already satisfied (use --upgrade to upgrade): cymem<1.32,>=1.30 in /usr/local
Requirement already satisfied (use --upgrade to upgrade): preshed<2.0.0,>=1.0.0 in /usr/l
```

```
Requirement already satisfied (use --upgrade to upgrade): thinc<6.11.0,>=6.10.1 in /usr/l
Requirement already satisfied (use --upgrade to upgrade): plac<1.0.0,>=0.9.6 in /usr/loca
Requirement already satisfied (use --upgrade to upgrade): pathlib in /usr/local/lib/pytho
Requirement already satisfied (use --upgrade to upgrade): ujson>=1.35 in /usr/local/lib/p
Requirement already satisfied (use --upgrade to upgrade): dill<0.3,>=0.2 in /usr/local/li
Requirement already satisfied (use --upgrade to upgrade): regex==2017.4.5 in /usr/local/l
Requirement already satisfied (use --upgrade to upgrade): wrapt in /usr/local/lib/python2
Requirement already satisfied (use --upgrade to upgrade): tqdm<5.0.0,>=4.10.0 in /usr/loc
Requirement already satisfied (use --upgrade to upgrade): cytoolz<0.9,>=0.8 in /usr/local
Requirement already satisfied (use --upgrade to upgrade): six<2.0.0,>=1.10.0 in /usr/lib/
Requirement already satisfied (use --upgrade to upgrade): termcolor in /usr/local/lib/pyt
Requirement already satisfied (use --upgrade to upgrade): msgpack-python in /usr/local/li
Requirement already satisfied (use --upgrade to upgrade): msgpack-numpy==0.4.1 in /usr/lo
Requirement already satisfied (use --upgrade to upgrade): toolz>=0.8.0 in /usr/local/lib/
You are using pip version 8.1.1, however version 10.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

which makes me believe that for some reason it was installed in the python2.7 of the root but I really don't get why. Maybe using `sudo` implicitly imply installing things on the root. So I tried

```
pip install spacy
```

and it seems to have worked!
But then I tried to do the next step of the tutorial but it didn't work:

```
(mypy36) aritz@aritz-ThinkPad-T460p:~$ python -m spacy.en.download all
/home/aritz/anaconda3/envs/mypy36/bin/python: Error while finding module specification fo
```

so instead I used

```
python -m spacy download en
```

and it worked. Then I could proceed with this tutorial:
`https://www.analyticsvidhya.com/blog/2017/04/natural-language-processing-made-easy-using`
`e2%80%8bin-python/`

From now on I will use always pip and not pip3 for my virtual environments.

I must say I don't really understand what are these "data and models" from spaCy. So I went on the official page:
`https://spacy.io/models/`
but it didn't help me much. I don't understand exactly what these models are supposed to do. It looks like spaCy can do plenty of things but it is not clear to me how this relates to pytorch and tensorflow. Is it supposed to be combined with these models or

can it perform these tasks completely independently?
It seems that there are two big libraries designed especially for NLP tasks: NLTK and spaCy. Here are some **difference between NLTK and spaCy**: From wikipedia: "Unlike NLTK, which is widely used for teaching and research, spaCy focuses on providing software for production usage." What comes again and again online is that NLTK provides more features, more liberty, and may be good for some specific rare tasks, and that spaCy just implements for each task the best method to perform it, with computationally more efficient way than NLTK. And the default methods of spaCy are supposed to be constantly updated to satisfy state of the art results, so updating spaCy might boost the performance of a script using it. So I guess that spaCy would be better for industry. These two pages provide summaries of the differences:
`https://blog.thedataincubator.com/2016/04/nltk-vs-spacy-natural-language-processing-in-p`
`https://www.reddit.com/r/LanguageTechnology/comments/69xbkc/question_spacy_or_nltk/`
From this last article the API's of spaCy are more object oriented which makes it more pythonic. The only real default of spaCy seems to be the fact that it was usable only for English at the time (April 2016) where the article was written. But this may have changed since then. When I go on this page:
`https://spacy.io/models/`
and look at the models it seems that other languages are supported.

I looked at the beginning of this tutorial:
`https://www.analyticsvidhya.com/blog/2017/04/natural-language-processing-made-easy-using-e2%80%8bin-python/`
which displays some of the possibilities of spaCy and copied the code in a notebook called spaCy_tutorial_trip_advisor_review and if I get it right, it is a library which does all these NLP tasks (without any help from TF or Pytorch (at least from the user perspective)). It seems very efficient and would save me a lot of time if I could use it instead of reimplementing it.

07.05.18 $\sim$ `string` **library, more on spaCy**
I discovered the string library which seems to contain plenty of useful utilities encapsulated in classes or constants. For instance `string.punctuation` is a string containing all the **punctuation symbols concatenated**:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

It was used in the Analytics Vidhya tutorial about spacy to **remove punctation from texts during the tokenization**.

I looked again at **spaCy**, using in particular this tutorial:
`https://www.analyticsvidhya.com/blog/2017/04/natural-language-processing-made-easy-using-e2%80%8bin-python/`
(this page `https://spacy.io/usage/linguistic-features` was quite useful to get the details).
I list here what I understood from it when going through the tutorial:

- Using this library and its "models" (for instance `en` as in `nlp = spacy.load("en")`) we can create objects from a string (which contains a text). Upon creation of this object, the text is parsed into token and sentences, tokens get various kinds of (predefined) labels/tags, entities are identified and assigned to some (predefined) labels/tags. All these quantities are stored as attributes of the object created (the type is `spacy.tokens.doc.Doc`) and can then easily be accessed.
- I have the impression that there are two ways to load a model:

  ```
  nlp = spacy.load("en")
  ```

  and

  ```
  from spacy.lang.en import English
  parser = English()
  ```

- It performs dependency parsing on the tokens (I'm not very familiar with this yet).
- If I understand it well, some models have been pretrained and are automatically used in order to create these objects.
- One can train his own model on his own data set:
  `https://spacy.io/usage/training`
- It offers visualization tools:
  `https://spacy.io/usage/visualizers`

This tutorial might also be good even if it dates from 2015:
`http://nicschrading.com/project/Intro-to-NLP-with-spaCy/`

Edit (30.11.18): In a script from Vien I ran into the class **PhraseMatcher** from spacy which allows one to find some patterns in a list of documents (or words) (a bit like regex) and can trigger some actions depending on what is found. This page gives a good idea of the basics:
`https://stackoverflow.com/questions/47638877/using-phrasematcher-in-spacy-to-find-multip`

17.05.18 ∼ `eval`**, and** `ast.literal_eval`**, Regular expressions,** `repr`
I learned a bit about the `eval` command on Stack overflow:
Stackoverflow: What does pythons eval do?
and also about `ast.literal_eval` which seems to be a safer version of `eval` (which should actually be avoided) here:
Stackoverflow: Using python's eval() vs ast.literal_eval()?

I encountered it in the notebook from the NLP coursera course "week1-MultilabelClassification", when we want to transform a string containing "`['php', 'mysql']`" into a python list, containing `'php'` and `'mysql'`.

I learned a bit about **regular expressions** here:
`https://www.regular-expressions.info/quickstart.html`
and how to use it in python here:
`https://docs.python.org/3.5/library/re.html#re.sub`
and in particular there:
`https://docs.python.org/3.5/howto/regex.html#regex-howto`

Nikos showed me how to **print the value of a string to see it clearly**. One has to use `repr`, which prints it as it is really (one see the spaces before and after, the backslashes used to protect a symbol and so on).

18.05.18 ∼ `string.strip`**,** `scipy.sparse`
The `strip` method of strings can be used to remove white spaces (by default) or any other characters possibly present at the beginning and end of a string. It is used in the notebook of the first week of coursera's course on NLP.

In the notebook of the first week of the NLP coursera's course, I discovered `scipy.sparse` which is the scipy class for **storing efficiently sparse matrices**. This can be useful for instance when one uses a bag of word method to represent a text in a numerical form. An interesting comment in the notebook is that sklearn algorithms can only work with csr matrices.

19.05.18 ∼ **Scikit-learn** `TfidfVectorizer` **tool for TF-IDF bag-of-word creation**
I learn how to build a bag of word dataset with tf-idf scores out of a list of strings with this tool from sklearn:
`http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`
It was used in the first notebook of the NLP coursera course.

24.05.18 ∼ **Scikit-learn tool for multi-label encoding**
I discovered this tool for creating a binary vector out of a list of label (typically the response variable a multi-label classification problem).

25.05.18 ∼ **F1-score for binary or multilabel classification**
In the notebook of the first week of the coursera NLP course, I (re)discovered a useful metric to judge binary or multilabel classification, namely the F1-score which has a nice scikit-learn implementation:

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

31.05.18 ∼ **Meaning in of `main`, underscore in Python**
I learned a bit about the meaning of the `main` API there:
https://stackoverflow.com/questions/4041238/why-use-def-main#4041253
and there
https://stackoverflow.com/questions/419163/what-does-if-name-main-do
It allows one to differentiate two usages of a .py script: the first is to execute the script, let say script1.py, (the part which will be put in the core of the `main` function), and the second is import it as a module from another script, let say script2.py, to be able to use some functions defined in script1.py.
I learned about **the meaning of the underscore** _ in Python there:
https://hackernoon.com/understanding-the-underscore-of-python-309d1a029edc

02.06.18 ∼ **Creating dictionaries dynamically**
I read about `collections.defaultdict` there:
Stack Overflow: How does collections.defaultdict work?
which allow to create dictionaries dynamically.

05.07.18 ∼ **Adding a dimension to an array**
I stumbled upon `numpy.newaxis` which allow to transform a 1D array into a 2D one, a 2D one into a 3D one etc... There is a nice explanation here:
Stack Overflow: How does numpy.newaxis work and when to use it?
It has an equivalent in TensorFlow called `tf.newaxis`.

03.08.18 ∼ **Unpacking with asterisk**
I learned another way to use the asterisk. I saw this first in the notebook of week 3 assignment of the NLP course:

```
for line in validation[:3]:
    q, *examples = line
    print(q, *examples[:3])
```

and I found an explanation here:
https://medium.com/understand-the-python/understanding-the-asterisk-of-python-8b9daaa4a5

06.08.18 ∼ **Reading lists from text with `ast.literal_eval`**

In the notebook of week 1 of the NLP class, I came across `ast.literal_eval` which is explained here:

`https://docs.python.org/3.5/library/ast.html#ast.literal_eval`
It allows one to create a python object out of a string like "['php', 'mysql']" (here we would create a list containing some strings which represent labels).

09.08.18 ~ Difference between & and `and`
I found out that there is a difference between & and `and`. This page explains it:
Difference between and boolean vs bitwise in python
The conclusion seems good to keep in mind:

- If you are not dealing with arrays and are not performing math manipulations of integers, you probably want `and`.
- If you have vectors of truth values that you wish to combine, use numpy with &.

15.08.18 ~ 'xsrf' argument missing from POST, modifying variables inside functions
I had already encountered the following problem: suddenly the message

```
'_xsrf' argument missing from POST
```

appears in the right corner of my jupyter notebook and I cannot save anything anymore. In fact it happened because I deleted my cookies.

This post on Stack Overflow made me realize plenty of things:
Why can a function modify some arguments as perceived by the caller but not others?
In particular one can modify a variable (or more specifically the value to which a name is referencing) defined outside the function with methods, typically:

```
my_l.append(4)
```

but if we use inside the function something like

```
my_l = [2, 4]
```

then we create a new variable/name which will disappear outside of the function.
This page:
`https://docs.python.org/3/reference/executionmodel.html#resolution-of-names`
explains why if one uses something like

```
a = 0
def f(x):
    b = a
    a = a + x
    return((a, b))
f(10)
```

returns an error of the type

```
UnboundLocalError: local variable 'a' referenced before assignment
```

To avoid this, the simplest is to pass a as a parameter to the function.
This answer also explains these stories of mutable and immutable objects, which are related to the question of changing the value of a variable inside a function:
Immutable vs mutable

19.08.18 ~ Reading csv files with Python, Counters in Python
I read this interesting blog about how to access .csv files in python:
`https://medium.com/district-data-labs/simple-csv-data-wrangling-with-python-3496aa5d0a5e`
There are a couple of interesting points:

1. Better to encapsulate the reading of the data into a function or a class.
2. Loading dataframes with pandas or numpy is not memory efficient, so should be avoided for big data frames.
3. Encoding issues and a library to avoid problems.
4. Using namedtuple to create new immutable types to represent the events/row. It looks like using a dictionary to represent a row but the difference is that one cannot change the entries (immutable type) and it is more memory efficient (faster access and three times memory to store it).
5. "how to serialize CSV data with Avro so that the data is stored not only in a compact format but with its schema as well". From what I understand, he present a better way to save the data than .csv file (more efficient from the memory perspective) and keeps information about the meaning of the different columns.

I learned about **counters in python** here:
`https://www.pythonforbeginners.com/collection/python-collections-counter`

11.09.18 ~ (Multiple) Class Inheritance with `super().__init__()`
I read this tutorial about **inheritance** in Python:
`https://www.python-course.eu/python3_inheritance.php`
I then read this second tutorial about **multiple inheritance** in Python:
`https://www.python-course.eu/python3_multiple_inheritance.php`
and learned the meaning of

```
super().__init__()
```

which appears (for instance) in sgdr.py of fastai.

14.09.18 ∼ **Debugging with `pdb`**
I learned the basics of debugging with `pdb` by reading this page:
`https://pythonconquerstheuniverse.wordpress.com/2009/09/10/debugging-in-python/`
It seems that the main purpose (or the most common one) is to display the value of some variables at some point of the execution.
(I encountered it in a tutorial about callbacks in fastai.

18.04.18 ∼ **Unit tests with `unittest`**
I learned about unit tests with `unittest` there:
`https://pythontesting.net/framework/unittest/unittest-introduction/`
and one can find an example from Bilyana there:
`https://ghe.exm-platform.com/Telepathy-Labs/tl_nlu_core/blob/master/tests/test_featurizers.py`

15.11.18 ∼ **How to make a (real) copy of a dict**
In order to copy a dictionary and have its content independent from the content of the original one, one should use the `copy` library as in

```
import copy
dict2 = copy.deepcopy(dict1)
```

the other solutions showed on this stack overflow post do not work for nested dictonaries.

29.11.18 ∼ **Warnings and errors in Python**
I learned how to **produce or filter warnings** in a python script, on this page:
`https://pymotw.com/2/warnings/`
and this one gives the full documentation:
`https://docs.python.org/3/library/warnings.html#module-warnings`
These two pages give a good introduction to **raising and handling errors** in python:
`https://stackoverflow.com/questions/2052390/manually-raising-throwing-an-exception-in-py24065533`
`https://en.wikibooks.org/wiki/Python_Programming/Errors`

11.02.19 ∼ **Log files for Python**

I came accross this tutorial teaching how to use **logs in Python**:
`https://fangpenlin.com/posts/2012/08/26/good-logging-practice-in-python/`
and this one also seemed good: `https://logmatic.io/blog/python-logging-with-json-steroids/`
I should read them entirely when I'll have the time/need for it.

This page explains what is the difference with warnings (showed in the entry above):
https://stackoverflow.com/questions/9595009/python-warnings-warn-vs-logging-warning
I find this part particularly instructive:

"warnings.warn() in library code if the issue is avoidable and the client application should be modified to eliminate the warning

logging.warning() if there is nothing the client application can do about the situation, but the event should still be noted"

27.03.19 ~ Unit test with pytest, subprocess module

It seems that there are several **testing tools in python**. I discovered unittest and pytest. I read this good tutorial about **unittest**:
https://pythontesting.net/framework/unittest/unittest-introduction/

I read this short tutorial introducing **pytest**:
https://semaphoreci.com/community/tutorials/testing-python-applications-with-pytest
Followed by this more complete tutorial, also about pytest (containing links to many other tutorials about testing):
https://pythontesting.net/framework/pytest/pytest-introduction/

From what I understand, pytest is easier to use and more general than unittest (since one can use scripts developed with unittest in the pytest framework). But they aren't that different. What is important to know is that the names of the test scripts should start by 'test_' (without the slash), that the name of the test function/methods should also start by 'test_', the name of the test classes should start with 'Test', as explained here:
https://pythontesting.net/framework/pytest/pytest-introduction/#discovery

I also read about software API/CLI adapters here:
https://pythontesting.net/strategy/software-api-cli-interface-adapters/
which enable to **test some python scripts from inside another python** script. This page was useful to understand the **subprocess module**:
https://www.bogotobogo.com/python/python_subprocess_module.php

02.04.19 ~ Static Type Checking

Since Python 3.6, one can use **static type checking**, meaning we can verify that variables and arguments have the correct type. This tutorial introduces it well:
https://medium.com/@ageitgey/learn-how-to-use-static-type-checking-in-python-3-6-in-10-

17.05.19 ∼ **Python headers**

Bilyana sent me a link toward this page of stack overflow concerning good practices for **python scripts headers**:
`https://stackoverflow.com/questions/1523427/what-is-the-common-header-format-of-python-f`

19.06.19 ∼ **Requests (HTTP library for Python)**

Roman showed me this page which gives an introduction to **request** the library for **http requests**:
`https://2.python-requests.org/en/master/user/quickstart/`
This library should be combined with the `urllib` which allows to put some data in a format which can then be used as part of the url address we use for a query like

```
quoted_question = urllib.parse.quote(question).replace('/', ' ')
```

and then

```
url = 'http://{}:{}/answer_from_text/{}/{}'.format(api_host, api_port, quoted_question, c
```

and

```
results = requests.get(url, headers=headers).json()
```

05.07.19 ∼ **Asynchronous programming,** `asyncio` **library**

I stumbled upon this piece of code in processor.py from Rasa (1.1.4):

```
    async def handle_message(
        self, message: UserMessage
    ) -> Optional[List[Dict[Text, Any]]]:
        """Handle a single message with this processor."""

        # preprocess message if necessary
        tracker = await self.log_message(message)
        if not tracker:
            return None

        await self._predict_and_execute_next_action(message, tracker)
        # save tracker state to continue conversation from this state
        self._save_tracker(tracker)

        if isinstance(message.output_channel, CollectingOutputChannel):
            return message.output_channel.messages
        else:
return None
```

(found here:
https://github.com/RasaHQ/rasa/blob/master/rasa/core/processor.py)

Because of this, I learned a bit about **asynchronous programming**. I read this wikipedia page:
https://en.wikipedia.org/wiki/Asynchronous_communication

I read this page which explains **what is asynchronous programming**:
https://www.quora.com/What-is-asynchronous-programming

I read this page which explains the **difference between asynchronous programming and parallel programming**:
https://stackoverflow.com/questions/6133574/how-to-articulate-the-difference-between-asy
6133756

Then I read this page which gives an **overview of asynchronous programming in python**:
https://medium.com/@nhumrich/asynchronous-python-45df84b82434
(but there were several parts which were a bit unclear even if the overall quality of the explanations is good).

Finally I read this tutorial which gives some examples of the `asyncio` library (the one at the end seem to have a problem because when I execute the script in a terminal, it doesn't exit):
https://stackabuse.com/python-async-await-tutorial/

Edit (18.11.19): I found this tutorial about the `asyncio` python library, which is very good as it seems complete:
https://realpython.com/async-io-python/
I read half of it (I stopped at the Async IO roots in generator section).

15.07.19 ~ Linting and Flake8

I discovered **Flake8** a library which helps to check your code syntax and provide instructions on how to clean it:
https://medium.com/python-pandemonium/what-is-flake8-and-why-we-should-use-it-b89bd78073

13.08.19 ~ spaCy meets PyTorch-Transformers: Fine-tune BERT, XLNet and GPT-2

It seems that the team of spaCy released a new library to use wrappers around pytorch based implementation of **Transformer, BERT, XLNet** and **GPT-2**:
https://explosion.ai/blog/spacy-pytorch-transformers

14.08.19 ~ python virtual environment with virtualenv

I think that `virtualenv` is a simple way to use python virtual environments.

Edit (01.10.19): This page gives a good summary of 3 techniques to create virtual environments:
`https://medium.com/@krishnaregmi/pipenv-vs-virtualenv-vs-conda-environment-3dde3f6869ed`

22.08.19 ~ Memory measurements with `memory_profiler` library

I came across `memory_profiler` library which allows to perform memory measurements with its `memory_usage` function.

27.08.19 ~ Good .gitignore for python

Rolland showed me this page where one can copy a good **.gitignore** for python based projects:
`https://github.com/github/gitignore/blob/master/Python.gitignore`

07.10.19 ~ How to write good python requirements.txt

I found this Quora answer pretty nice, concerning ways to write **requirements.txt**:
`https://www.quora.com/How-do-you-make-a-requirements-txt-file-for-Python-Is-the-version-`

Edit (01.11.19): I found these two links which give some insight concerning the way requirements files are done, and what are the **different ways one can list required libraries** depending on the type of libraries:
`https://pip.readthedocs.io/en/1.1/requirements.html`
`https://pip-python3.readthedocs.io/en/latest/reference/pip_install.html#vcs-support`
In particular I understood that when a package is on the pypi (list of python packages available for everyone and light enough to be put there), we can just put something like

```
Keras==2.2.4
```

whereas if it is a heavy package, it will have to be downloaded from another source, and could look like this

```
https://github.com/keras-team/keras-contrib/archive/e1574a16f0156dd9557fe62a1c5a39e37bd65
```

and if its some private library in a private repos, then it will look like

```
git+https://ghe.exm-platform.com/MyPrivateRepo/my_project.git@master#egg=my_project
```

09.10.19 ~ setup.py

The **setup.py** script is a package included in libraries that is downloaded when we do 'pip install ...', which tells the system how to install the given library (dependencies etc...).

06.11.19 ~ `sanic`: a python web server and web framework

I found in rasa code, reference to the sanic library:
`https://sanic.readthedocs.io/en/latest/`
"The goal of the project is to provide a simple way to get up and running a highly performant HTTP server that is easy to build, to expand, and ultimately to scale."
It is supposed to be simple to use.

Edit (03.01.20): I read this small tutorial which explains how to run a server on your local machine using sanic, **ngrok**, and a trial account with **Twilio**:
`https://www.twilio.com/blog/2016/12/getting-started-with-sanic-the-asynchronous-uvloop-b`
`html`
Note that at the moment where I write this, python 3.5 is not enough (unlike what is written in the tutorial) and at python 3.6 is the minimum required.
I also read this page from the sanic doc which explains how to configure your server:
`https://sanic.readthedocs.io/en/latest/sanic/config.html`

Edit (05.01.20): From what I understand, one of the main things that sanic does is to map predefined types of http requests (by default, I think that it is assumed to be a GET request, which just means that the client tries to access some data from the server) to some python functions which do an action when a client sends the given request. The mapping is done using decorators. Here is a simple example where the server is then run locally (I'm still confused about how to run a server and where to run it):

```
from sanic import Sanic
from sanic.response import text


app = Sanic()


@app.route("/")
async def hello(request):
    return text("Hello World!")

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)
```

Note that there exists other similar decorators, like `@app.get` or `@app.post` which are wrappers for `@app.route` with methods argument taking the corresponding value like:

```
@app.route('/get', methods=['GET'], host='example.com')
async def get_handler(request):
    return text('GET request - {}'.format(request.args))
```

12.11.19 ∼ **Namespace**

I read an introduction to **Namespaces**, here:
`https://code.tutsplus.com/tutorials/what-are-python-namespaces-and-why-are-they-needed--`

03.01.20 ∼ **Personal voice assistant in python**

I found this simple voice assistant all in python which can be a good baseline for a virtual voice assistant:
`https://www.geeksforgeeks.org/personal-voice-assistant-in-python/`

26.11.19 ∼ **Click library**

I discovered the click library which can be used to create **command line tools**:
`https://click.palletsprojects.com/en/7.x/`

06.01.20 ∼ **Text preprocessing**

I found this Kaggle kernel for **text preprocessing**
`https://www.kaggle.com/sudalairajkumar/getting-started-with-text-preprocessing`
which shows easy ways to do all operations like:

- lower casing
- removal of punctuation
- removal of stopwords
- removal of frequent/rare words
- stemming and lemmatization
- removal of emojis/emoticons
- conversion of emoticons/emojis to words

- removal of URLs
- removal of HTML tags
- chat words conversion
- spelling correction

The data is customer services conversations on twitter.

07.01.19 ~ Text classification

This kaggle kernel presents several classical ways to do **text classification**:
`https://www.kaggle.com/abhishek/approaching-almost-any-nlp-problem-on-kaggle`
It includes

- tfidf
- count features
- logistic regression
- naive bayes
- svm
- xgboost
- grid search
- word vectors
- LSTM
- GRU
- Ensembling

24.04.20 ~ Fask and Blueprints

Since a while I'm trying to understand the concept of **blueprints** which is implemented in the sanic library used by rasa. Here is the piece of code inside rasa/core/channels/channel.py where it appears:

```
def register(
    input_channels: List["InputChannel"], app: Sanic, route: Optional[Text]
) -> None:
    async def handler(*args, **kwargs):
```

```
        await app.agent.handle_message(*args, **kwargs)

    for channel in input_channels:
        if route:
            p = urljoin(route, channel.url_prefix())
        else:
            p = None
        app.blueprint(channel.blueprint(handler), url_prefix=p)

    app.input_channels = input_channels
```

I didn't find any good explanation of what a sanic blueprint is but it seems to be a concept appearing in many web application framework, and I found some documentation about the **flask** blueprints:
`http://exploreflask.com/en/latest/blueprints.html`
From I understand, the role of a flask (and probably sanic) app object is to map a request made at a certain url (the two together form a **route** from what I understand) to a python function. **Blueprints** help mapping a group of similar request to a same function. For instance if several URLs have a the same prefix (facebook/profile/pictures.html and facebook/profile/friends) we might want to put them in a similar blueprint.
The concept of **view** comes also frequently. From what I understand it is a python function associated to a route.

I also learned about **Web Server Gateway Interface (WSGI)** which is something specific to python but related to web application.

Here is the docstring of the `Blueprint` class of the **sanic** library:

"In *Sanic* terminology, a **Blueprint** is a logical collection of URLs that perform a specific set of tasks which can be identified by a unique name."

12.05.20 ~ Custom Word2Vec embedding with gensim

I created some custom word embeddings with **gensim**.
An important note is that it was recommended to install **cython** in the python environment to make the training efficient.

The **preprocessing**, is actually not necessarily trivial. Steps like removing stop words and punctuation on the sentences which will be used as training material for the Word2Vec are meaningful if one wants to use embedding vectors for similarity measures but not necessarily if one wants to feed the word vectors to a model making some more complex task than finding what is similar to what. Lemmatization is also something good for word similarity but not necessarily for the rest. It might be also useful to

replace alpha-numerical quantities by something tokens which do not vary (replace phone numbers by phone_number for instance).
This page give me some good indications:
https://www.quora.com/Why-should-punctuation-be-removed-in-Word2vec
Here is the pipeline I used pipeline:

1. Lower case text
2. Replace entity values by entity types
3. Split into sentences
4. Remove punctuation
5. Remove stop words
6. Lemmatize
7. Map remaining sequences of numbers to "number_"

Using the following links:
https://machinelearningmastery.com/develop-word-embeddings-python-gensim/
https://www.analyticsvidhya.com/blog/2019/07/how-to-build-recommendation-system-word2vec
https://radimrehurek.com/gensim/models/word2vec.html

I wrote this little **script to train** the Word2Vec model and find words similar to a given one:

```
import os
import json
from gensim.models import Word2Vec
from gensim.test.utils import get_tmpfile



def main():
    # Load sentences and split them into seperate tokens
    training_sentences_fn = os.path.join(os.getcwd(), 'word_emb_training_sentences.json')
    with open(training_sentences_fn, mode='r', encoding='utf-8') as f:
        sentences = json.load(f)

    path = get_tmpfile("word2vec.model")

    # Train model
    model = Word2Vec(size=100, window=5, min_count=3, workers=4, seed=666)

    model.build_vocab(sentences, progress_per=200)
```

```
    model.train(sentences, total_examples=model.corpus_count,
                epochs=10, report_delay=1)

    model.save("word2vec.model")

    model.init_sims(replace=True)

    print(model)

    vector = model.wv['replace']

    most_similar = model.similar_by_vector(vector, topn=7)[1:]

    new_ms = []
    for j in most_similar:
        pair = (j[0], j[1])
        new_ms.append(pair)

    print(new_ms)


if __name__ == '__main__':
    main()
```

Edit (26.05.20): If one wants to use these embeddings to find sentences (in a data set of sentences (most likely the same used for training the Word2Vec model)), one can either do the computation "manually" using `sklearn.metrics.pairwise.cosine_similarity` and some sorting algorithm, or use another tool from gensim: `gensim.similarities.SoftCosineSimilari` In this second case, the code looks like this:

```
word_vectors_fn = os.path.join(os.getcwd(), 'vectors.kv')
    word_vectors = KeyedVectors.load(word_vectors_fn, mmap='r')

    similarity_index = WordEmbeddingSimilarityIndex(word_vectors)
    similarity_matrix = SparseTermSimilarityMatrix(similarity_index, dictionary)

    query_bow = tfidf[dictionary.doc2bow(query_preprocessed)]
    index = SoftCosineSimilarity(
        tfidf[[dictionary.doc2bow(sent) for sent in sentences_li]],
        similarity_matrix)

    similarities = index[query_bow]
```

```
sim_with_index = [(i, sim) for i, sim in enumerate(similarities)]
sim_with_index_sorted = sorted(sim_with_index, reverse=True, key=lambda el: el[1])
```

but in both cases it is very **slow** with one sentence to compare to around 40K sentences it takes around 40 seconds. I found it better to use a simple Bag Of Word with TF-IDF and similar gensim tool for that. The code then looks like

```
query_bow = tfidf[dictionary.doc2bow(query_preprocessed)]

index_tmpfile = get_tmpfile("index")
index = Similarity(
    output_prefix=index_tmpfile,
    corpus=tfidf[[dictionary.doc2bow(sent) for sent in sentences_li]],
    num_features=len(dictionary))

similarities = index[query_bow]
```

4 Questions

1. If I want to check that the arguments received by a function are of the appropriate type and throw an error if they aren't how do I do?

5 Useful How-Tos

1. **Create a virtual environment**:
   There are several methods to do that. One of the easiest and most powerful way is to use conda, and create a virtual environment with the desired version of python and a list of libraries like this

   ```
   conda create --name environment_name python=3.7
   ```