Cours TAL - Laboratoire 1a - Mise en place de l'environnement

Bienvenue au premier labo du cours TAL !

Pour travailler avec ce *noteboook*, il faut d'abord télécharger l'archive ZIP du Laboratoire 1 qui inclut ce *notebook* en PDF et les fichiers `TAL_Labo1a.ipynb`, `TAL_Labo1b.ipynb`, et `TAL_Labo1c.ipynb`. Vous pouvez commencer par lire le fichier PDF avant de mettre en place l'option qui vous conviendra parmi les suivantes, ce qui vous permettra d'ouvrir les fichiers `.ipynb`.

Résumé

Le but de cette première partie (a) du Laboratoire 1 est de vous permettre d'installer et de tester l'environnement de programmation qui vous servira tout au long du Cours TAL. Le langage de programmation sera Python 3 avec les *notebooks* Jupyter qui vous permettront d'écrire des instructions, conserver leurs résultats, et ajouter vos commentaires.

Dans les parties (b) et (c) de ce Laboratoire 1 (*notebooks* appelés `TAL_Labo1b.ipynb` et `TAL_Labo1c.ipynb`), vous utiliserez la boîte à outils NLTK pour travailler sur des fichiers locaux ou en ligne, et spécifiquement les segmenter en phrases et en mots.

Pour soumettre vos résultats, veuillez exécuter toutes les cellules de chaque *notebook*, enregistrer les *notebooks* en ajoutant vos noms au nom du fichier, puis regrouper les deux fichiers `TAL_Labo1b_NOM1_NOM2.ipynb` et `TAL_Labo1c_NOM1_NOM2.ipynb` dans un fichier `zip`, qui sera rendu sur Cyberlearn.

Options d'utilisation de Python et Jupyter

Tout d'abord, décidez si vous utiliserez un gestionnaire d'environnements comme Conda -- utile par exemple si vous devez gérer plusieurs versions incompatibles de Python ou de ses *packages* sur une même machine. Si oui, veuillez suivre les instructions de l'option 1 ci-dessous, puis continuez avec l'option 2. Sinon, passez directement à l'option 2. Si vous ne souhaitez pas du tout travailler en local, suivez l'option 3. En résumé :

- travailler sur votre ordinateur ou sur un service en ligne ?
  - ordinateur (recommandé) : *passez à la question suivante*
  - en ligne: *option 3*
- environnement unique ou multiple ?
  - unique (plus simple) : *option 2* seule
  - multiple (recommandé car plus flexible) : *option 1* puis *option 2*

Option 1 : gestionnaire d'environnements Conda et notebooks Jupyter

Avant d'installer Python et Jupyter sur votre ordinateur, vous pouvez installer un gestionnaire d'environnements qui évitera les conflits entre différentes versions de Python et des ses *packages* dont vous pourriez avoir besoin. Par exemple, vous pouvez installer le gestionnaire Anaconda / Conda / Miniconda. Si vous choisissez Miniconda3, Python sera inclus dans l'installation. Anaconda est une distribution plus grande qui contient beaucoup de *packages* Python installés.

Instructions pour installer Conda

- Suivez les instructions à https://conda.io/projects/conda/en/latest/user-guide/install/index.html (suggestion : considérez l'installation de Miniconda3).
- Les programmes d'installation (exécutable Windows, script `bash` pour Linux) sont à https://conda.io/en/latest/miniconda.html -- par exemple, sous Linux, exécuter `wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh` puis

`bash Miniconda3-latest-Linux-x86_64.sh` .

- Après installation, on appelle Conda en ligne de commande, dans un `command prompt` fourni par Conda. Pour une introduction très rapide à Conda, voir https://conda.io/projects/conda/en/latest/user-guide/getting-started.html et pour une liste de commandes : https://conda.io/projects/conda/en/latest/user-guide/cheatsheet.html.
- Pour créer l'environnement dédié au cours de TAL : `conda create --name CoursTAL python=3` -- puis pour l'activer : `source activate CoursTAL` .
- Noter la commande `conda list` pour voir les pacakages installés par défaut, et la commande `source deactivate` pour quitter l'environnement.

Instructions pour utiliser Jupyter dans un environnement Conda

Jupyter est un serveur qui supporte des pages dynamiques avec des textes, du code, et des résultats. On peut demander à Jupyter d'utiliser un environnement Conda comme *kernel*. Installez Jupyter (voir Option 2) dans l'environnement du `CoursTAL` puis exécutez la commande : `python -m ipykernel install --user --name CoursTAL --display-name "CoursTAL"` . Cela permet de choisir pour un *notebook* Jupyter l'environnement "CoursTAL" comme kernel.

Autre option : créer l'environnement et y ajouter Jupyter en une seule commande : `conda create -n CoursTAL python=3 ipykernel ipywidgets` .

Option 2: installation sans Conda de Python et Jupyter

- Installez un interpréteur Python 3 (typiquement la dernière version).
- Installez ensuite le package Jupyter. Vous pouvez utiliser le Package Installer for Python (pip) pour ajouter de nouveaux *packages* (en général avec la commande `pip install xxxxx`).
  - Ces opérations se font en ligne de commande (sous Windows: `cmd` ou Power Shell).
- Créez un dossier où vos stockerez vos *notebooks* et ajoutez-y les *notebooks* `TAL_Labo1a.ipynb` , `TAL_Labo1b.ipynb` , et `TAL_Labo1c.ipynb` .
- Depuis le dossier précédent (`cd xxxx`), lancez le serveur Jupyter avec la `jupyter notebook --no-browser`
- Ouvrez l'URL http://localhost:8888 pour voir la liste des *notebooks* connus de Jupyter, puis exécutez celui appelé `TAL_Labo1a.ipynb` -- celui-ci même.

Option 3 : travailler sur un service en ligne (Colab)

Vous pouvez utiliser l'environnement en ligne gratuit proposé par Google, appelé Colab. Vous aurez toutefois besoin d'un compte Google (par exemple Gmail) et vos fichiers seront stockés sur Google Drive. Colab est offert par Google pour expérimenter avec le *machine learning* (sur CPU, GPU ou TPU), mais pas pour des calculs très intenses.

Une fois votre session ouverte sur Colab, vous devrez importer les *notebooks* Jupyter vers cet Colab. Les *packages* Python que nous utiliserons sont déjà installés.

Quelques tests simples

Pour tester votre installation, veuillez écrire un simple calcul dans la cellule suivante, par exemple calculez combien de séquences de trois lettres on peut former avec les 26 lettres (minuscules) de l'alphabet latin (écrire les multiplications).

In [1]:
```
# Ecrivez votre code à la ligne qui suit, puis exécutez la cellule pour afficher le résultat
```

Pour vérifier que vous pouvez écrire quelques lignes (très simples) en Python, créez un tableau contenant les entiers de 1 à 10, puis affichez la liste de leurs valeurs au carré. Vous pouvez vous inspirer de ce Tutoriel Python de l'université d'Edimbourg.

In [2]:
```
# Ecrivez votre code à la ligne qui suit, puis exécutez la cellule pour afficher le résultat
```

Comment se familiariser avec Python?

Comme indiqué ci-dessus, la première option est le tutoriel de l'UEdin qui est proposé dans le cours de TAL de l'UEdin. Ce cours offre aussi une bonne introduction au *package* `numpy` . Leur premier *notebook* introduit aussi Matplotlib, qui permet de faire des graphiques dans Jupyter.

Une introduction systématique au langage Python est proposée par J.R. Johansson dans ses Scientific Python Lectures, qui sont accompagnées de *notebooks* Jupyter. La documentation Python en ligne est également utile.

Fin de la partie (a) du Laboratoire 1

Veuillez passer maintenant au *notebook* `TAL_Labo1b.ipynb` . Ce notebook `TAL_Labo1a.ipynb` n'est pas à rendre.