Formation Python

KI '022

Ecole des Ponts Paristech

14 septembre 2020

Sommaire

Introduction

Pour commencer il faut :

- Avoir Python sur son ordinateur
- Avoir un IDE
- Avoir un gestionnaire de paquet

Plusieurs options :

- Télécharger Python depuis leur site `https://www.python.org/downloads/`
- (mieux) Utiliser Anaconda/Miniconda
- (Linux) `sudo apt install python3.8`

Si vous en avez déjà un gardez-le (sauf si c'est Pyzo...) mais il y en a plein de très bien :

- **PyCharm :** Excellent pour les gros projets, super correction syntaxique je vous le recommande !
- **Atom :** Similaire à PyCharm, mais n'est pas restreint à Python.
- **VsCode :** Plus léger que Pycharm, il utilise aussi moins de ressources, et possède une très bonne correction syntaxique.

On peut aussi utiliser des éditeurs de texte comme SublimeText ou Notepad++ pour des petits projets, cela fait l'affaire.

Avoir un gestionnaire de paquets

Pour cette étape il y a deux options :

- Utiliser conda (recommandé)
- Utiliser pip (Déjà fournit avec les verions de Python > 3.4)

Télécharger les fichiers

La plupart du temps vous aurez besoin d'ouvrir un fichier existant et de le modifier. Dans le cadre de la formation rendez-vous sur `https://github.com/KIClubinfo/TP_python` et téléchargez le dossier (ou avec git clone...).

Lecture de fichier

Dans le TP vous avez un fichier data.txt qu'il vous faudra lire et traiter.

```
1 fichier = open("data.txt", "r")
2 lignes = fichier.readlines()
3 L =[]
4 for ligne in lignes:
5     L.append(ligne.strip("\n").split(","))
6 fichier.close()
```

Explications

- `open` : Ouvre le fichier, le `"r"` signifie qu'on veut juste lire le fichier (on ne le modifie pas)
- `.readlines()` : Met chaque ligne du fichier dans une liste.
- `.strip()` : retire les caractères choisis (ici les marque de fin de lignes *i.e* les `\n`)
- `.split()` : sépare la chaîne de caractère en fonction du caractère passé en argument (ici les donnés sont séparées par une virgule)

Les bonnes habitudes

- Toujours fermer le fichier une fois qu'on a récupéré les donnés, quitte à le ré ouvrir.
- `readlines()` va lire tout le fichier, c'est à dire qu'un curseur se déplace pour lire chaque élément, ainsi si vous refaites `fichier.readlines()` vous n'aurez que du vide car le curseur sera déjà à la fin du fichier.
- Il existe aussi `.readline()` qui va juste lire une ligne et déplacer le curseur à la fin de celle-ci.

A vous !

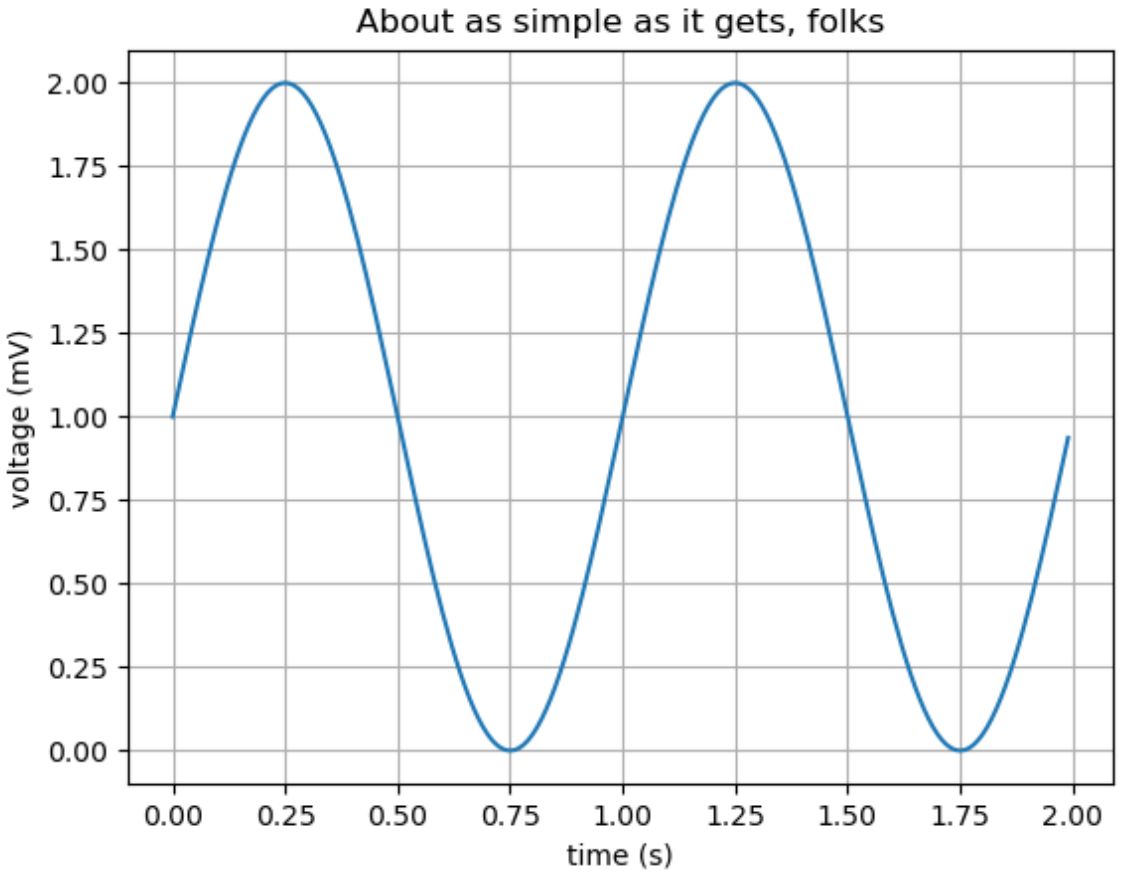
C'est à vous, essayez de lire le fichier `"data.txt"` afin d'avoir une liste de liste de flottants.

Le package Matplotlib

Vous connaissez sûrement ce package pour tracer des courbes/graphes issus de vos donnés.

```
1  import matplotlib
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  # Data for plotting
6  t = np.arange(0.0, 2.0, 0.01)
7  s = 1 + np.sin(2 * np.pi * t)
8
9  fig, ax = plt.subplots()
10 ax.plot(t, s)
11
12 ax.set(xlabel='time (s)', ylabel='voltage (mV)',
13 title='About as simple as it gets, folks')
14 ax.grid()
15
16 fig.savefig("test.png")
17 plt.show()
```

Le package Matplotlib



https://matplotlib.org/tutorials/index.html

Tracer ses résultats

Nous avons une liste de donnés qui consiste en 2 lignes de flottants, nous allons tracer la courbe qui a pour abscisse la première ligne et la deuxième en ordonné.

- Importer `matplotlib.pyplot`
- Tracer la courbe correspondant aux donnés de `data.txt`
- Tracer la courbe de la fonction exponentielle en échelle log

A vous !

- La fonction `plot` pour tracer
- Les fonctions `title, xlabel, ylabel` pour le noms des axes et le titre
- La fonction `show()` pour afficher

Parfois vos résultats auront plus de sens si vous les visualisez en échelle log, par exemple pour trouver une constante k tel que $\|f_n - f\|_\infty \leq C^k$ (coucou le TP d'AnaCs).

Pour vous entrainer, essayer de tracer la fonction exponentielle en échelle log en ajoutant `plt.yscale('log')` dans votre code.

$\rightarrow$ Conseil : utilisez la fonction `arange` du module `numpy`.

Listes en compréhension

Les listes Python sont très souples et permettent beaucoup de choses par exemple si vous voulez les carrés de chaque nombre pair entre 0 et 10 vous pourvez écrire :

```
1 L = []
2 for k in range(11):
3     if k%2 == 0:
4         L.append(k**2)
```

Ou alors :

```
1 L = [k for k in range(11) if k%2 ==0]
```

Attention à l'affectation !

Imaginons que vous vouliez travailler sur une liste `L` sans la modifier, par exemple retirer certains de ses éléments. La première approche consisterait à faire `G = L` puis de travailler sur la liste `G`.

Attention à l'affectation !

Imaginons que vous vouliez travailler sur une liste `L` sans la modifier, par exemple retirer certains de ses éléments. La première approche consisterait à faire `G = L` puis de travailler sur la liste `G`.

Oui mais en fait non ! Le problème avec ceci c'est que si vous modifiez `G` vous allez aussi modifier `L`. En fait `G = L` signifie littéralement "`G` est la même liste que `L`"

Le "slicing"

Il vous sera parfois utile de sélectioner seulement une partie d'une liste, c'est à dire tout sauf les 3 derniers éléments par exemple ou encore un élément sur deux... Cela s'appelle le "slicing" voyons quelques exemples.

Le "slicing"

```
1 L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3 L[1:5]
4 L[:3]
5 L[:6:2]
6 L[:-3]
7 L[-2::-1]
```

Renvoie :

```
1 [2, 3, 4, 5]
2 [1, 2, 3]
3 [1, 3, 5]
4 [1, 2, 3, 4, 5, 6, 7]
5 [9, 8, 7, 6, 5, 4, 3, 2, 1]
```