Introduction à la Programmation Python

Microprojet

- Objectif pédagogique
- Exercice

Formation permanente du CNRS, Délégation Alsace
Février - Mars 2017

Auteurs :

- Vincent Legoll (vincent.legoll@iphc.cnrs.fr)
- Matthieu Boileau (matthieu.boileau@math.unistra.fr)

Objectif pédagogique

- Utiliser les modules de la librairie standard pour récupérer des données via un service web.
- Manipuler les dictionnaires et les chaînes de caractères
- Utiliser la librairie de tracés graphiques matplotlib
- Utiliser un IDE (Spyder)
- Exécution en script
- Gestion des arguments

Exercice

Exploiter les données du site http://www.prevision-meteo.ch pour tracer l'évolution horaire de la température à Strasbourg aujourd'hui.

Ouverture du fichier de prévisions

Le site http://www.prevision-meteo.ch fournit des prévisions sous forme de fichier au format json. On veut récupérer les données relatives à Strasbourg avec la méthode `urlopen()` du module `urllib.request`.

```
In [1]: from urllib.request import urlopen

        jsonfile_url = "http://www.prevision-meteo.ch/services/json/Strasbourg"
        f = urlopen(jsonfile_url)  # open url
```

Chargement du fichier json ouvert

La méthode `json.loads()` permet de charger un fichier json comme un dictionnaire python :

```
In [2]: import json
        jsonfile = json.loads(f.read().decode("utf-8"))  # Read JSON file
```

Exploration des données

On commence naïvement par afficher le contenu du fichier :

```
In [3]: print(jsonfile)
```

```
{'current_condition': {'humidity': 63, 'date': '16.03.2017', 'wnd_spd': 1, 'hour':
```

On essaie de faire mieux en affichant uniquement les clés du dictionnaire :

```
In [4]: for k in jsonfile:
            print(k)
```

```
current_condition
fcst_day_3
fcst_day_1
```

```
fcst_day_2
fcst_day_4
fcst_day_0
city_info
forecast_info
```

On est intéressé par le temps d'aujourd'hui :

```
In [5]: day = jsonfile['fcst_day_0']
        print(day)
```

```
{'icon_big': 'http://www.prevision-meteo.ch/style/images/icon/ensoleille-big.png',
```

Là aussi, on cherche les clés :

```
In [6]: for k in day:
            print(k)
```

```
icon_big
condition
date
hourly_data
icon
tmax
tmin
condition_key
day_short
day_long
```

Vérifions qu'il s'agit d'aujourd'hui :

```
In [7]: print(day['day_long'], day['date'])
```

```
Jeudi 16.03.2017
```

C'est bon ! Maintenant, une entrée particulière nous intéresse :

```
In [8]: day_hd = day['hourly_data']
        for k in day_hd:
            print(k)
```

```
1H00
17H00
5H00
10H00
2H00
```

```
0H00
14H00
7H00
19H00
15H00
13H00
9H00
18H00
3H00
6H00
4H00
8H00
11H00
21H00
20H00
23H00
16H00
22H00
12H00
```

Regardons ce que contient une 'hourly_data' :

```
In [9]: for k in day_hd['8H00']:
            print(k)
```

```
WNDGUST10m
KINDEX
CONDITION
WNDSPD10m
HUMIDEX
WNDDIR10m
CONDITION_KEY
PRMSL
APCPsfc
CAPE180_0
HCDC
HGT0C
TMP2m
MCDC
DPT2m
RH2m
LCDC
WNDCHILL2m
ICON
ISSNOW
CIN180_0
WNDDIRCARD10
```

La clé qui nous intéresse est la chaîne `'TMP2m'` qui correspond à la température à 2m du sol.

```
In [10]: hour = '12H00'
         print("Aujourd'hui à {}, il fera : {} deg. C.".format(hour, day_hd[hour]['

Aujourd'hui à 12H00, il fera : 14.3 deg. C.
```

Sauver ces lignes de commandes dans le fichier `today_stras.py` en allant de l'exécution 1 au compteur d'exécution courant indiqué dans la cellule de code ci-dessus `In [XX]`. Dans le cas présent :

```
In [11]: %save today_stras.py 1-10

The following commands were written to file `today_stras.py`:
from urllib.request import urlopen

jsonfile_url = "http://www.prevision-meteo.ch/services/json/Strasbourg"
f = urlopen(jsonfile_url)  # open url
import json
jsonfile = json.loads(f.read().decode("utf-8"))  # Read JSON file
print(jsonfile)
for k in jsonfile:
    print(k)
day = jsonfile['fcst_day_0']
print(day)
for k in day:
    print(k)
print(day['day_long'], day['date'])
day_hd = day['hourly_data']
for k in day_hd:
    print(k)
for k in day_hd['8H00']:
    print(k)
hour = '12H00'
print("Aujourd'hui à {}, il fera : {} deg. C.".format(hour, day_hd[hour]['TMP2m']))
```

Tracé de la température

1. Ouvrir le fichier `today_stras.py` dans Spyder et nettoyer les `print` inutiles.
2. Exécutez le code dans Spyder et utilisez la fenêtre "Variable explorer" en haut à droite pour parcourir les données de votre dictionnaire.
3. Extraire la liste des couples `(hour, temperature)` où :
   - `hour` est un entier
   - `temperature` est un flottant
4. ordonner la liste selon les heures croissantes

5. convertir la liste en un *numpy array* `t` avec la méthode `numpy.array()`
6. Transposer `t` pour obtenir le tableau `[[liste of hours], [list of temperatures]]`
7. réaliser un tracé matplotlib en suivant ce tutoriel ou en intégrant les lignes de code suivantes :

```
In [12]: import matplotlib.pyplot as plt  # To be placed at the top of python file

         # [Your previous code...]

         # Plot T = T(hour)
         fig = plt.figure()  # initialise figure
         title = "{} {}".format(day_of_the_week, date_of_today)
         fig.suptitle(title, fontsize=14, fontweight='bold')

         ax = fig.add_subplot(111)  # initialise a plot area
         fig.subplots_adjust(top=0.85)
         ax.set_title('Day temperature')
         ax.set_xlabel('Time [h]')
         ax.set_ylabel('Temperature [deg. C]')

         ax.plot(t[0], t[1])  # plot t[1] (tempe) as a function of t[0] (hour)
```

```
/opt/conda/lib/python3.5/site-packages/matplotlib/font_manager.py:273: UserWarning:
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take
/opt/conda/lib/python3.5/site-packages/matplotlib/font_manager.py:273: UserWarning:
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-12-c01ce31f4615> in <module>()
      5 # Plot T = T(hour)
      6 fig = plt.figure()  # initialise figure
----> 7 title = "{} {}".format(day_of_the_week, date_of_today)
      8 fig.suptitle(title, fontsize=14, fontweight='bold')
      9


NameError: name 'day_of_the_week' is not defined
```

```
<matplotlib.figure.Figure at 0x7f9ca415b048>
```

Option : intégrer l'icone de la météo du jour en utilisant le module `matplotlib.image`

Solution dans exos/meteo_json.py

Exercice : Modifiez le programme météo en créant une fonction qui admet un des jours disponibles comme argument (aujourd'hui, demain, après-demain...)

Une proposition de solution dans exos/meteo_json_func.py

Exécution en script

Pour pouvoir exécuter ce fichier en mode script

- Ajouter en première ligne du fichier: `#! python3`
- Rendez le fichier exécutable: `chmod a+x today_stras.py`

Pour différencier du code devant s'exécuter en mode script (par opposition a un import du module) on utilise la variable `__name__` pour exécuter du code différent.

```
In [13]: #! python3

         def main():
             print('je suis dans un script')

         if __name__ == '__main__':
             main()
         else:
             # En mode module importé, on ne fait rien de plus
             pass
```

```
je suis dans un script
```

Gestion des arguments

Pour pouvoir passer des arguments en ligne de commande, le module `argparse` peut etre utilisé, entre autres (`getopt`, etc.).

Un tutoriel plus complet existe ici.

```
In [14]: import argparse

         parser = argparse.ArgumentParser(description='Process some integers.')
         parser.add_argument('integers', metavar='N', type=int, nargs='+',
                             help='an integer for the accumulator')
         parser.add_argument('--sum', dest='accumulate', action='store_const',
                             const=sum, default=max,
                             help='sum the integers (default: find the max)')
         print(type(parser))
```

```
<class 'argparse.ArgumentParser'>
```

```
In [15]: # Sans "--sum"
         args = parser.parse_args(['2', '5'])
         print(args.accumulate(args.integers))
```

```
5
```

```
In [16]: # Avec "--sum"
         args = parser.parse_args(['--sum', '2', '5'])
         print(args.accumulate(args.integers))
```

```
7
```

```
In [17]: # Certains arguments existent déjà
         try:
             parser.parse_args(['--help'])
         except SystemExit:
             # Pour éviter une erreur dans jupyter-notebook
             pass
```

```
usage: __main__.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
  N           an integer for the accumulator

optional arguments:
  -h, --help  show this help message and exit
  --sum       sum the integers (default: find the max)
```

Exercice : Modifiez le programme météo pour qu'il prenne le(s) nom(s) de ville en argument(s) en utilisant le module `argparse`

Une proposition de solution dans exos/meteo_json_func_args.py

Suite de l'exercice

- Laissez libre cours à vos idées et envies, par exemple :
  - en cherchant à tracer l'évolution horaire de la température dans les 5 prochains jours
  - etc.
- Dans Spyder :
  - testez le système de debugging
  - testez le profiler

À vous de faire la pluie et le beau temps !