Test mid-term SIE-BA3 ENG-270

Scientific Python (Numpy, Matplotlib), Programmation Objet, Git

Samuel Bancal

2020-10-30

Contents

1 Mesure d'humidité dans le sol d'un verger

1.1 Contexte

Il s'agit de grapher et mettre en lumière des données d'humidité du sol mesurées dans un verger dans le canton de Genève. L'objectif initial de ces mesures est d'offrir à l'exploitant un moyen de monitorer en direct le taux d'humidité et ainsi planifier l'arrosage du verger grâce à l'installation existante.

Nous allons ici simplement revenir sur les mesures des mois de juin, juillet et août 2020, grapher les courbes des mesures d'humidité dans le sol au pied d'un arbre et repérer avec des couleurs l'état d'arrosage correspondant.

Trois senseurs sont placés au pied de cet arbre, chacun à 30 cm de profondeur. Le premier à 1m à gauche, le second à 1 cm et le troisième à 1m à droite du point de chute de l'arrosage.

Les données mesurées sont en [kPa] et pour le sol concerné il est fixé les seuils suivants, avec les couleurs que nous choisissons pour les mettre en lumière sur les graphs :

[kPa]	État de l'arosage	Couleur dans le graph	Label
0 - 15	Saturé	`'red'`, `alpha=0.2`	`'saturated'`
15 - 30	Trop humide	`'orange'`, `alpha=0.2`	`'too wet'`
30 - 60	Suffisament humide	`'green'`, `alpha=0.2`	`'perfect'`
60 - 100	Gamme d'irrigation	`'yellow'`, `alpha=0.2`	`'plan to water'`
100 - 200	Sec	`'red'`, `alpha=0.2`	`'dry'`

1.2 La source de données

Nous allons travailler avec des données brutes, c-à-d non formatées ou arrangée pour cet exercice. Il s'agira donc de récupérer ces données par un téléchargement, puis dans notre code *Python*, d'en extraire les données utiles à notre traitement.

Il faudra également transformer certaines données erronées. En effet, sur la plage de temps que nous travaillons, certains capteurs n'ont pas répondu, ce qui se reconnaît avec des valeurs saturées à 200. Il faudra donc remplacer toutes ces valeurs saturées par une "non-valeur" ... un trou.

2 Les fichiers attendus

Vous allez rendre un dossier nommé `test_2020-10-30`, versionné avec Git, répliqué sur Github, comprenant 6 fchiers :

1. `my_pledge.md`
   Fichier dont le contenu vous est donné ci-dessous. Il décrit l'engagement que vous prenez en passant ce test.
2. `eco-sensors_irrigation_2020-06-01_2020-08-31.json`
   Fichier que vous téléchargerez ici :
   https://enacit.epfl.ch/cours/docs/eco-sensors_irrigation_2020-06-01_2020-08-31.json
   Il contient les données brutes qu'il faudra transformer et grapher.
3. `process_irrigation.py`
   Ce fichier contiendra le code *Python* que vous aurez écrit pour ateindre l'objectif décrit ci-dessous.
4. `irrigation_graph_2020-06.png`
   Image au format PNG, généré par votre code Python couvrant le mois de juin 2020. L'exemple du résultat attendu est montré ci-dessous.
5. `irrigation_graph_2020-07.png`
   Image au format PNG, généré par votre code Python couvrant le mois de juillet 2020. L'exemple du résultat attendu est montré ci-dessous.
6. `irrigation_graph_2020-08.png`
   Image au format PNG, généré par votre code Python couvrant le mois d'août 2020. L'exemple du résultat attendu est montré ci-dessous.

3 Travail versionné avec Git

L'entier du travail devra être versionné avec Git et déposé sur un dépôt privé hébergé sur Github.

- Chaque commit devra être fait avec comme auteur: votre nom, prénom et email.
- Le dépôt Github, nommé `test_2020-10-30`, devra être privé et vous devrez me donner les droits pour que je puisse récupérer votre travail. Pour cela, rendez-vous sur la page principale de votre dépôt sur github *https://github.com/your_username/test_2020-10-30* . Dans l'onglet (en haut à droite) *Settings* > volet (à gauche) Manage access, choisissez *[Invite a collaborator]*, cherchez l'utilisateur "sbancal", puis confirmez avec *[Add sbancal to this repository]*.

3.1 Le 1er commit

Intitulé "*Mon engagement*", il contiendra exclusivement le fichier `'my_pledge.md'` dont le contenu sera le suivant :

```
# Engagement

Je m'engage à passer cet examen sans faire appel à une personne externe,
excepté les assistants et le professeur de ce cours.
Que ce soit oralement ou par tout autre moyen de communication.

votre Prénom Nom
```

3.2 le 2ème commit

Intitulé "*Ajout des mesures d'humidité*", incluera le fichier `'eco-sensors_irrigation_2020-06-01_2020-08-31.json'` que vous aurez téléchargé.

3.3 Les commits suivants

Chaque étape significative franchie dans le développement de votre solution sera accompagnée d'un *commit* avec un commentaire explicite.

4 Le code Python attendu

L'objectif du code que vous allez écrire est de :

- Lire les données mesurées depuis le fichier `'eco-sensors_irrigation_2020-06-01_2020-08-31.json'`
- Utiliser la librairie `json` pour transformer ces données sous forme de fichier texte en une structure de données *Python* dans un dictionnaire.
- Utiliser la librairie `pandas` pour transformer les séries de données en un `DataFrame`
  Comme nous n'avons pas utilisé cette librairie en cours, je vous fournirai les instructions nécessaires à cette étape.
- Nettoyer les données reçues avec l'appel de la fonction `clean_data(data)` que vous devrez écrire.
  Cette fonction remplacera toutes les valeurs saturées `200` par la valeur `np.nan` ... *Not a Number*. Cela créera dans les graphs des trous aux endroits où nous considérons qu'il n'y a aucune mesure viable.
- Générer les graphs demandés avec un appel par fichier à la fonction `save_plot_to_file(dataframe, title, labels, start_date, end_date, filename)` que vous devrez écrire.
  Cette fonction :
  - créera une figure de 10x10 pouces, en 100dpi
  - subdivisera la zone de graph en 3 subplots (utilisez la méthode `fig.subplots(...)` pour cela). Chacun des subplots aura :
    * l'axe des X qui sera commun
    * une trace du taux d'humidité mesuré dans le sol
    * les zones délimitées par les seuils décrits ci-dessus (utilisez la méthode `ax.fill_between` pour colorier ces zones)

* les `yticks` placés au milieu des seuils (positions à calculer soi-même) avec des labels tels que donnés dans le tableau ci-dessus
* l'axe Y limité explicitement entre `0` et `200`
* l'axe X limité entre la 1ère et la dernière date; sans marge supplémentaire
* sa légnde placée dans le coin en haut à gauche

On placera le titre au dessus du premier subplot et les dates sur l'axe X seront mises en forme automatiquement avec l'appel à `fig.autofmt_xdate()`

5 Structure du code Python attendu

Votre code devra être structuré en 5 portions distinctes et successives :

1. Documentation globale du code réalisé dans le fichier
2. Import des librairies nécessaires
3. Définition des `CONSTANTES` utiles pour l'ensemble du code, au minimum :
   - le nom du fichier JSON lu
   - les niveaux des différentes zones et leurs attributs
4. Définition des fonction suivantes :
   1. `clean_data(data)`
   2. `save_plot_to_file(dataframe, title, labels, start_date, end_date, filename)`

   Chacune de ces fonction devra être documentée, tel que `help(nom_de_la_fonction)` nous permette de connaître ce qu'elle fait.

   Si cela vous aide, vous pouvez créer plus de fonctions.
5. Définitiion du code principal dans un bloc `if __name__ == '__main__':`
   Il commencera par l'import des données provenant du fichier JSON (en 2 étapes: JSON -> dictionnaire, dictionnaire -> DataFrame Pandas), puis appelera `clean_data(...)` et finira avec les appels à `save_plot_to_file(...)` pour chaque graph désiré.

6 Structure du fichier JSON

Le fichier que nous avons reçu est plutôt volumineux et on pourrait vite passer du temps à chercher sa structure pour savoir quoi extraire. Pour cela, je vous le décrit ici de façon simplifiée (en retirant les parties inutiles) et vous indique les données à récupérer.

En le lisant, vous remarquerez une liste de plusieurs dictionnaires. Seuls les 3 premiers nous seront utiles (le 4ème, inutile pour ce travail, n'est pas documenté ici).

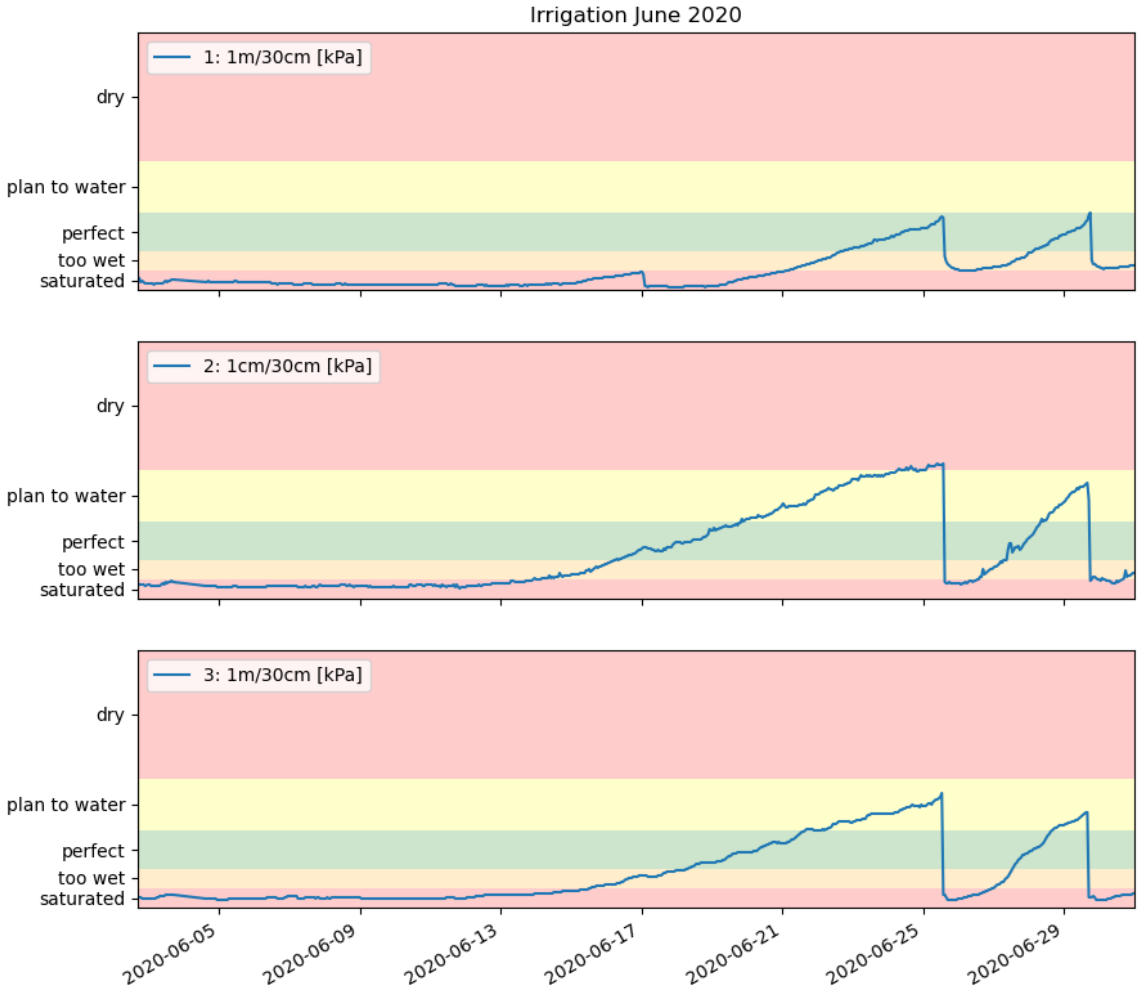
En prévision des graphs que nous voulons faire nous avons besoin :

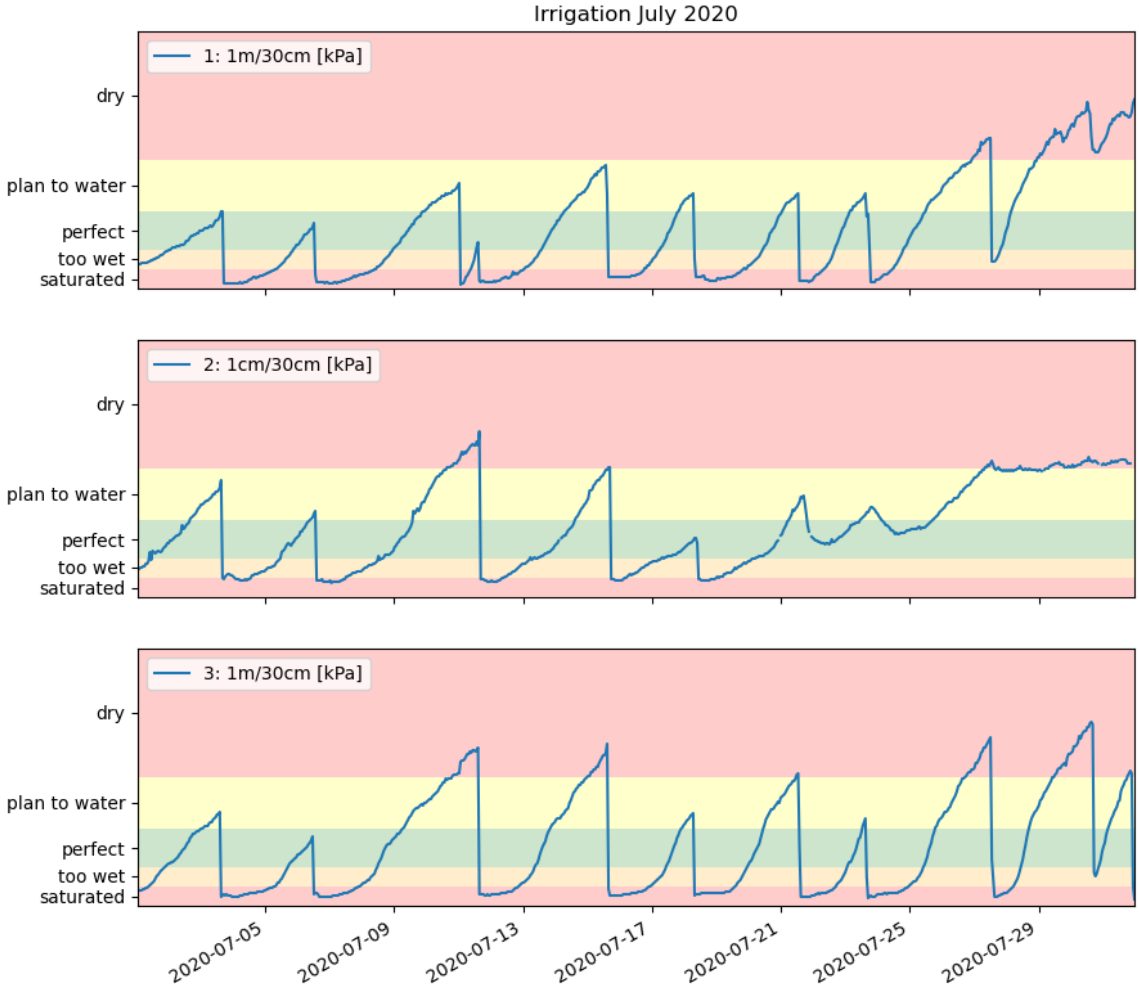
1. d'une série décrivant le temps. Elle nous sera donnée par la liste `[num]['labels']`. Cette série est répétée à l'identique dans chaque dictionnaire. Une seule nous suffira.
2. des labels décrivant les 3 senseurs. Ils nous seront donnés par les valeurs `[num]['datasets']['label']`
3. des valeurs mesurées par les 3 senseurs. Elles nous seront données par les listes `[num]['datasets']['data']`

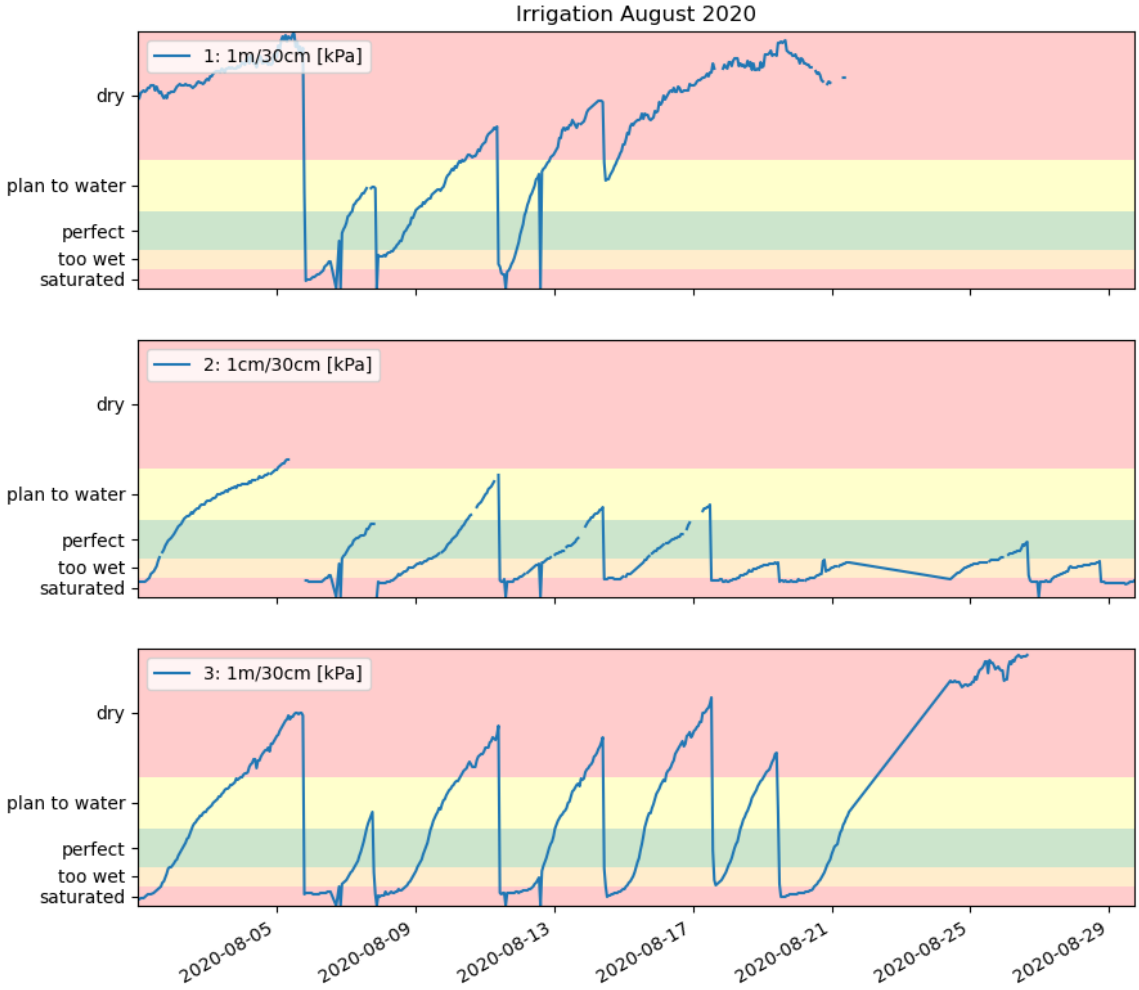

```
[
    {
        "labels": [ "2020-06-02 16:48:28", ... ],
        "datasets": {
            "label": "1: 1m/30cm [kPa]",
            "data": [ 10, ... ]
        },
        "unit": "kPa", "station": "Station 1 (Rang\u00e9e 8)"
    },
    {
        "labels": [ "2020-06-02 16:48:28", ... ],
        "datasets": {
            "label": "2: 1cm/30cm [kPa]",
            "data": [ 12, ... ]
        },
        "unit": "kPa", "station": "Station 1 (Rang\u00e9e 8)"
    },
    {
        "labels": [ "2020-06-02 16:48:28", ... ],
        "datasets": {
            "label": "3: 1m/30cm [kPa]",
            "data": [ 8, ... ]
        },
        "unit": "kPa", "station": "Station 1 (Rang\u00e9e 8)"
    }
]
```

7 Le résultat

Voici les 3 fichiers que vous devez reproduire :




Irrigation July 2020
1: 1m/30cm [kPa]
dry
plan to water
perfect
too wet
saturated
2: 1cm/30cm [kPa]
dry
plan to water
perfect
too wet
saturated
3: 1m/30cm [kPa]
dry
plan to water
perfect
too wet
saturated
2020-07-05
2020-07-09
2020-07-13
2020-07-17
2020-07-21
2020-07-25
2020-07-29


Irrigation August 2020
1: 1m/30cm [kPa]
2: 1cm/30cm [kPa]
3: 1m/30cm [kPa]
dry
plan to water
perfect
too wet
saturated
2020-08-05
2020-08-09
2020-08-13
2020-08-17
2020-08-21
2020-08-25
2020-08-29

8 Aide

8.1 la librairie `json`

Utilisez simplement le code suivant pour charger dans `json_data` le contenu du fichier JSON préalablement ouvert en lecture.

```
import json

json_data = json.load(filehandler)
```

8.2 la librairie `pandas`

8.2.1 Création de l'objet `DataFrame`

Utilisez et adaptez le code suivant pour créer un objet DataFrame qui contiendra toutes les valeurs requises à l'exercice. Vous devrez remplacer `label1`, `label2`, `label3`, `data1`, `data2`, `data3` et `time_index` par les vraies valeurs.

```
import pandas as pd

humidity_dataframe = pd.DataFrame(
    data={
        label1: data1,
        label2: data2,
        label3: data3,
    },
    index=time_index,
    dtype='float'
)
humidity_dataframe.index = pd.to_datetime(humidity_dataframe.index)
```

Vous pouvez observer le contenu de ce `DataFrame` avec plusieurs méthodes tel que :

```
print(humidity_dataframe)
#                      1: 1m/30cm [kPa]  2: 1cm/30cm [kPa]  3: 1m/30cm [kPa]
# 2020-06-02 16:48:28              10.0               12.0               8.0
# 2020-06-02 17:48:34               9.0               11.0               8.0
# 2020-06-02 18:48:40               6.0               11.0               8.0
# 2020-06-02 19:48:46               7.0               11.0               7.0
# 2020-06-02 20:48:53               6.0               11.0               7.0
# ...                               ...                ...               ...
# 2020-08-29 13:22:52             200.0               11.0             200.0
# 2020-08-29 14:22:58             200.0               12.0             200.0
# 2020-08-29 15:23:04             200.0               12.0             200.0
# 2020-08-29 16:23:10             200.0               12.0             200.0
# 2020-08-29 17:23:16             200.0               13.0             200.0
#
# [1999 rows x 3 columns]

print(humidity_dataframe.index)
# DatetimeIndex(['2020-06-02 16:48:28', '2020-06-02 17:48:34',
#                '2020-06-02 18:48:40', '2020-06-02 19:48:46',
#                '2020-06-02 20:48:53', '2020-06-02 21:48:59',
#                '2020-06-02 23:49:11', '2020-06-03 00:49:17',
#                '2020-06-03 01:49:23', '2020-06-03 02:49:30',
#                ...
#                '2020-08-29 07:22:15', '2020-08-29 08:22:21',
#                '2020-08-29 09:22:27', '2020-08-29 10:22:33',
#                '2020-08-29 11:22:39', '2020-08-29 13:22:52',
#                '2020-08-29 14:22:58', '2020-08-29 15:23:04',
#                '2020-08-29 16:23:10', '2020-08-29 17:23:16'],
#              dtype='datetime64[ns]', length=1999, freq=None)
```

```
print(humidity_dataframe.describe())
#        1: 1m/30cm [kPa]  2: 1cm/30cm [kPa]  3: 1m/30cm [kPa]
# count       1999.000000        1999.000000       1999.000000
# mean          64.572586          46.989795         53.060330
# std           66.272302          40.219993         52.094793
# min            0.100000           0.100000          0.100000
# 25%            9.000000          15.000000         10.000000
# 50%           35.000000          34.000000         34.000000
# 75%          112.500000          69.000000         79.000000
# max          200.000000         200.000000        200.000000

print(humidity_dataframe.dtypes)
# 1: 1m/30cm [kPa]     float64
# 2: 1cm/30cm [kPa]    float64
# 3: 1m/30cm [kPa]     float64
# dtype: object
```

8.2.2 Extraction depuis l'objet `DataFrame`

Utilisez et adaptez le code suivant pour extraire un jeu de données pour une période donnée. L'objet `values` ici obtenu est de type *numpy.ndarray*

```
start_date = '2020-06-01'
end_date = '2020-06-30'
values = humidity_dataframe[start_date:end_date][label1].values
```

Pour extraire l'indexe (le temps représenté sur l'axe X), utilisez ceci :

```
start_date = '2020-06-01'
end_date = '2020-06-30'
values = humidity_dataframe[start_date:end_date].index
```

9 Soumission de votre copie

9.1 Voie normale - Github.com

Note : voie uniquement possible pour ceux qui :

1. m'ont communiqué leur username Github par moodle
2. m'ont donné accès à leur dépôt durant l'examen selon la procédure décrite plus haut.

Je récupérerai votre travail à l'heure annoncée depuis votre dépôt Github.

Une fois le travail fini, n'oubliez pas de vérifier que tous les fichiers sont *commités* (y.c. les image PNG) à la dernière version et que vous avec *pushé* tout ça sur Github! ; -)

9.2 Voie alternative - Moodle.epfl.ch

Si vous ne parvenez pas à versionner votre travail avec Git ou n'arrivez pas à l'envoyer sur Github, vous devez préparer un Zip du dossier dédié à cet examen et l'envoyer dans l'espace sur Moodle prévu à cet effet.

10 Barème sur 9.0 points

Points	Sujet
1.0	**Rendre une copie** sur Moodle ou Github.com
—	—
2.0	**Git**
0.25	Dépôt Git initialisé
0.25	Auteur correctement configuré
0.25	2 premiers commits tel que demandés
0.5	Commits aux étapes significatives
0.5	Messages des commits compréhensibles et décrivant ce qui est fait
0.25	Dépôt sur Github.com correctement configuré
—	—
6.0	**Python**
0.5	Syntaxe OK
0.25	Le code s'exécute sans erreur
0.25	Code structuré en 5 portions tel que demandé
0.5	Définition des constantes utiles (minimum 2)
0.25	Lecture des données avec `json`
0.25	Lecture des données avec `pandas`
	Fonction `clean_data`
0.25	Aide en ligne
0.25	fonctionne comme attendu
0.25	écrit avec `numpy` (efficace)
	Fonction `save_plot_to_file`
0.25	Aide en ligne
0.25	figure tel que demandé
0.25	avec 3 subplots
0.25	subplots avec axe X synchro
0.25	titre, légendes
0.25	traces des mesures d'humidité
0.25	ticks en X
0.5	ticks en Y
0.25	limites en X
0.25	limites en Y
0.5	coloration des 5 zones

Le calcul de la note se fera ainsi : $NoteFinale = \frac{7.5 * PointsObtenus}{9.0}$

Dès que vous rendez votre copie : $NoteFinaleMin = 1.0$

Et naturellement : $NoteFinaleMax = 6.0$