

Projets de développement informatique : Labo 1

Github et Travis

1 Contexte

Le but de ce labo consiste à réaliser un projet simple qu'il faudra héberger sur la plateforme GitHub et ensuite tester à l'aide de la plateforme Travis. Pour rappel, ces deux plateformes sont disponibles aux adresses suivantes :

- GitHub : <https://github.com>
- Travis : <https://travis-ci.org/>

L'idée consiste à produire un module qui va proposer plusieurs outils de calculs mathématiques. On va donc devoir commencer par créer le module Python composé de plusieurs fonctions de calcul. On s'assurera que ce module est correct en écrivant des tests unitaires à faire vérifier par Travis.

2 Installation de Git et mise en place du dépôt

1. Installer `git`¹.
2. Configurer `git` sur sa machine (il ne faut le faire qu'une seule fois). Lancez un terminal (ou GitBash sur Windows) et entrez les commandes suivantes (avec la même adresse e-mail que celle utilisée sur GitHub) :

```
1 $ git config --global user.name "Sébastien Combéfis"
2 $ git config --global user.email "seb478@gmail.com"
```

Aller sur
github; compte et
créer sur le site
internet

3. Créez un nouveau dépôt sur GitHub (New repository) nommé `AdvancedPython2BA-Labo1`
4. Récupérez une copie locale du dépôt sur votre machine en créant un clone, avec la commande suivante (en reprenant l'URL de votre dépôt GitHub évidemment) :

```
1 $ git clone https://github.com/<votreNom>/AdvancedPython2BA-Labo1.git
```

5. Une fois cela fait, vous êtes prêts à travailler sur le projet. Vous allez pouvoir créer, modifier, supprimer des fichiers et créer des commits. De temps en temps, n'oubliez pas d'envoyer vos commits sur le dépôt distant, il vous suffit pour cela d'exécuter la commande suivante (qui vous demandera vos identifiants GitHub) :

```
1 $ git push origin master
```

3 Définition de la librairie `utils.py`

Reprenez le fichier `utils.py` de base proposé sur Claco. Ajoutez-le dans votre dossier de travail et faites un premier commit avec uniquement ce fichier ajouté, en mettant comme message "Initial Commit". Comme vous l'aurez remarqué, ce fichier ne contient que des fonctions non définies ainsi qu'un code principal qui permet d'exécuter ce fichier.

Une fois ce premier commit fait, créez une classe de test pour ce module. Repartez pour cela du squelette fourni sur Claco, et écrivez un test par fonction. Une fois cela fait, lancez le fichier et les tests

1. Un guide d'installation se trouve ici : <http://sebold.combefis.be/files/trainings/GIT-form-2014-Installation.pdf>

devraient tous échouer. Si cela fonctionne, créez un nouveau commit, en mettant comme message “Add unit test for utils module”.

On va maintenant configurer le Travis. Pour cela, il faut ajouter un fichier `.travis.yml` dans votre dossier, que vous trouverez sur Claco. Une fois cela fait, créez un nouveau commit avec comme message “Add Travis configuration”.

À ce stade, vous pourriez aussi avoir un dossier `__pycache__` qui a été créé dans votre dossier. Il ne faut évidemment pas le maintenir dans le dépôt Git. On peut signaler à Git qu’il faut l’ignorer en ajoutant un fichier `.gitignore` comme celui posté sur Claco. Ajoutez-le à votre dossier et créez un nouveau commit avec comme message “Add .gitignore file”.

4 Tests unitaires et configuration de Travis

Rendez-vous maintenant sur le site de Travis et créez-y vous y un compte. Il va ensuite falloir lier votre compte GitHub pour que Travis puisse avoir accès à vos dépôts. Ensuite, il faudra activer les dépôts pour lesquels vous souhaitez que Travis exécute les tests.

Si tout a bien été fait, vous devriez donc avoir vos tests qui échouent et donc recevoir un e-mail le signalant. Corrigez maintenant le module `utils` pour que les tests passent. Une fois cela validé en local, faites un nouveau commit avec la nouvelle version du fichier `utils` et avec comme message “Fix utils module to pass tests”.

Comparez ensuite vos tests avec les tests d’autres étudiants, et complétez les vôtres afin qu’ils soient le plus complet possibles. Créez un commit avec votre fichier `test_utils.py` mis à jour avec comme message “Add more tests”. Corrigez ensuite l’implémentation du module `utils` si jamais elle ne passe plus les tests, et faites un commit avec de nouveau comme message “Fix utils module to pass tests”.

5 Consignes générales et astuces

- Préférez faire pleins de petits commits qu’un seul gros de temps en temps. En règle générale, si vous n’êtes pas capable de trouver un court message descriptif du commit, c’est que vous tentez de mettre trop de changements dans le même commit.
- Ce n’est pas utile de **push** vos commits sur GitHub après chaque commit. Ne le faites que lorsque vous avez besoin que vos commits soient stockés sur le dépôt distant.
- Par convention, les messages de commit doivent être en anglais et toujours écrit sous la forme impérative.
- Python propose une fonction prédéfinie `eval` qui permet d’évaluer une expression Python.
- Le module `scipy` contient des fonctions pouvant vous aider pour l’intégrale. N’oubliez pas de mettre votre `requirements.txt` à jour.
- Pensez à présenter votre projet dans le `README.md`
- Avant de quitter le labo, postez sur claco un lien vers votre profil GitHub.

Bon travail !

