ECAM
Brussels Engineering School

jeudi 8 février 2018
CBF, LUR, LRG

Projets de développement informatique : Labo 1

Github, Heroku et Travis

1 Contexte

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

— GitHub : https://github.com
— Heroku : https://www.heroku.com
— Travis : https://travis-ci.org/

L'idée consiste à produire une application web qui va proposer plusieurs outils de calculs mathématiques. On va donc devoir commencer par créer un 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. On développera ensuite une simple application web grâce au module CherryPy, permettant d'effectuer les calculs proposés par le module précédemment développé. Ce dernier sera déployé sur Heroku.

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) :

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

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) :

```
$ git clone https://github.com/combefis/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) :

```
$ 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.

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

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 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 Interface web et déploiement sur Heroku

On va maintenant développer l'interface web qui va servir d'interface pour utiliser les fonctions mathématiques définies dans le module `utils`, et ensuite déployer l'application sur la plateforme Heroku. Commencez par récupérer les fichiers `server.py` et `server.conf` depuis Claco, contenant un squelette d'application web avec le module CherryPy. Ajoutez-les dans votre dossier et créez un commit avec comme message "`Add hello world cherrypy-based web server`".

On va maintenant mettre l'application en ligne via Heroku. Pour cela, il nous faut ajouter des fichiers de configuration à notre projet. Récupérez les fichiers `runtime.txt`, `requirements.txt` et `Procfile` depuis Claco et placez-les dans votre dossier. Créez ensuite un commit avec comme message "`Add Heroku configuration files`". Rendez-vous ensuite sur le site de Heroku et créez-y vous un compte. Une fois cela fait, vous devrez créer une nouvelle application et y associer votre compte GitHub. Il ne reste plus qu'à déployer l'application (bouton `Deploy Branch` tout en bas) et vérifier que tout fonctionne bien en ligne.

6 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. On peut par exemple écrire le code suivant qui affiche 14 après exécution.

```
1 x = 12
2 print(eval('x + 2'))
```

Maintenant, c'est à vous de jouer, et en mode freelance !

Commencez par le calcul de la factorielle d'un nombre naturel. Réalisez une page sur votre application web, vers laquelle un lien sera placé sur la page `home`, contenant un formulaire permettant d'entrer un nombre naturel, dont la factorielle sera calculée par le module `utils` puis affichée à l'écran.

Ensuite, faites une page calculant les racines d'un trinôme du second degré, et enfin celle pour le calcul d'une approximation de l'intégrale d'une fonction.

Avant de quitter le labo, envoyez par e-mail à `cbf@ecam.be` un lien vers votre profil GitHub.

Bon travail !