W1-S3-C1-utiliser-les-notebooks

December 15, 2014

1 "Notebooks" IPython comme support de cours

Pour illustrer les vidéos du MOOC, nous avons choisi d'utiliser IPython pour vous rédiger les documents "mixtes" contenant du texte et du code python, qu'on appelle des "notebooks", et dont le présent document est un exemple.

Nous allons dans la suite utiliser du code Python, pourtant nous n'avons pas encore abordé le langage. Pas d'inquiétude, ce code est uniquement destiné à valider le fonctionnement des notebooks, vous comprendrez ce code dans les semaines à venir. Cependant, vous constaterez sans doute que ces fragments de code sont facile à comprendre.

1.0.1 Avantages des notebooks

Comme vous le voyez, ce support permet un format plus lisible que des commentaires dans un fichier de code.

Les fragments de code peuvent être évalués et modifiés.

Notez bien également que le code python est interprété **sur une machine distante**, ce qui vous permet de faire vos premiers pas avant même d'avoir procédé à l'installation de python sur votre propre ordinateur.

1.0.2 Comment utiliser les notebooks

En haut du notebook, vous avez une barre, contenant d'abord un titre, le menu `File` ... `Kernel`, et une barre de boutons qui sont des raccourcis vers certains menus fréquemment utilisés. Si vous laissez votre souris au dessus d'un bouton, un petit texte apparaîtra, indiquant à quelle fonction correspond ce bouton.

Nous avons vu dans la vidéo qu'un notebook est constitué d'une suite de cellules, soit textuelles, soit contenant du code. Les cellules de code sont facilement reconnaissables, elles sont précédées de `In [ ]:`. La cellule qui suit celle que vous êtes en train de lire est une cellule de code.

Pour voir le résultat d'une cellule de code, il faut la sélectionner; puis vous disposez d'un bouton dans la barre des boutons en forme de flêche triangulaire vers la droite (Play), qui vous fait passer d'une cellule à la suivante, en évaluant le code avec un interpréteur Python qui s'appelle IPython.

Alternativement vous pouvez simplement taper au clavier ***Shift+Enter***, ou selon les claviers ***Maj-Entrée***, pour obtenir le même effet. La façon habituelle d'*exécuter* l'ensemble du notebook consiste à partir de la première cellule, et à taper ***Shift+Enter*** jusqu'au bout du notebook.

Entrainez-vous avec la cellule de code suivante:

```
In []: 20 * 30
```

Lorsqu'une cellule de code a été évaluée, IPython ajoute sous la cellule `In` une cellule `Out` qui donne le résultat du fragment python, soit ci-dessus 600.

IPython ajoute également un nombre entre les crochets pour afficher, par exemple ci-dessus, `In [1]:`. Ce nombre vous permet de retrouver l'ordre dans lequel les cellules ont été évaluées.

Vous pouvez naturellement modifier ces cellules de code pour faire des essais; ainsi vous pouvez vous servir du modèle ci-dessous pour calculer la racine carrée de 3, ou essayer la fonction sur un nombre négatif et voir comment est signalée l'erreur:

```
In []: # math.sqrt (pour square root) calcule la racine carrée
       import math
       math.sqrt(2)
```

On peut également évaluer tout le notebook en une seule fois en utilisant le menu *Cell -> Run All*

1.0.3 Attention à bien évaluer les cellules dans l'ordre

Il est important que les cellules de code soient évaluées dans le bon ordre. Si vous ne respectez pas l'ordre dans lequel les cellules de code sont présentées, le résultat peut être inattendu.

En fait, évaluer un programme sous forme de notebook revient à le découper en petits fragments, et si on exécute ces fragments dans le désordre, on obtient naturellement un programme différent.

On le voit sur cet exemple

```
In []: message = "Il faut faire attention à l'ordre dans lequel on évalue les notebooks"
```

```
In []: print message
```

Si un peu plus loin dans le notebook on fait par exemple

```
In []: del message
```

qui rend le symbole "message" indéfini, alors bien sûr on ne peut plus évaluer la cellule qui fait `print` puisque la variable `message` n'est plus connue de l'interpréteur.

1.0.4 Réinitialiser l'interpréteur

Si vous faites trop de modifications, ou perdez le fil de ce que vous avez evalué, il peut être utile de redémarrer votre interpréteur. Le menu *Kernel -> Restart* vous permet de faire cela, un peu à la manière de IDLE qui repart d'un interpréteur vierge lorsque vous utilisez la fonction F5.

Le menu *Kernel -> Interrupt* peut être quant à lui utilisé si votre fragment prend trop longtemps à s'exécuter (par exemple vous avez écrit une boucle dont la logique est cassée et qui ne termine pas)

1.0.5 Workflow des notebooks - Annuler les modifications

Un des avantages principaux des notebooks est de vous permettre de modifier le code que nous avons écrit, et de voir par vous mêmes comment se comporte le code modifié.

Pour cette raison chaque élève dispose de sa **propre copie** de chaque notebook, vous pouvez bien sûr apporter toutes les modifications que vous souhaitez à vos notebooks sans affecter les autres étudiants.

Vous pouvez toujours revenir à la version "du cours" grâce au menu

```
File -> Reset from Origin
```

Attention, avec cette fonction vous restaurez **tout le notebook** et donc **vous perdez vos modifications précédentes**.

1.0.6 Télécharger au format python

Vous pouvez télécharger un notebook au format python sur votre ordinateur grâce au menu

```
File -> Download as -> python
```

Les cellules de texte sont préservées dans le résultat sous forme de commentaires python

1.0.7 Partager un notebook en lecture seule

Enfin avec le menu

```
File -> Share static version
```

vous pouvez publier une version en lecture seule de votre notebook; vous obtenez une URL que vous pouvez publier par exemple pour demander de l'aide sur le forum; ainsi, les autres étudiants peuvent accéder en lecture seule à votre code.

1.0.8 Ajouter des cellules

Lorsque vous arrivez à la fin du document, une nouvelle cellule est créée chaque fois que vous évaluez la dernière cellule; de cette façon vous disposez d'un brouillon pour vos propres essais.

À vous de jouer.