Le Bestiaire de l'Algomathique

Quelques remarques en vrac

Ce que je pense ;-)

Je pense que chaque logiciel a son utilité :

⇒ Python permet de tracer des histogrammes, gérer des stats MAIS GeoGebra aussi et est souvent plus convivial (ou la calculatrice)
⇒ Python permet de faire du calcul formel, MAIS Xcas ou les modules de résolution de système de la calculatrice sont plus adaptés
⇒ Faire attention à la valeur ajoutée / temps d'apprentissage !

1. Documents

Beaucoup de documents m'ont inspirés / aidé pour la préparation de ce stage dont :

- ceux du site EduPython : https://edupython.tuxfamily.org/ MAIS je ne pense pas que ce soit une bonne idée d'utiliser le module lycee.
- le livre de Vincent Maille : *Python, les bases de l'algorithmique et de la programmation*.
- le livre de Gérard Swinnen : *Apprendre à programmer avec Python 3*. Existe en version papier et en version .pdf gratuite : https://inforef.be/swi/download/apprendre_python3_5.pdf
- le travail des collègues de l'académie de Créteil http://maths.ac-creteil.fr
- des fiches de résumé : https://perso.limsi.fr/pointal/python:memento

Quelques remarques en vrac

Importance du code

recommandations de la PEP 8 : https://www.python.org/dev/peps/pep-0008/

Un code est plus souvent lu qu'écrit ! Commenter les lignes par # et décrire les fonctions à l'aide de """..""" (trois double quotes).

Modules

Modules, Librairies, Bibliothèques

pour les importer :

```
1 from math import sqrt # permet l'accès à la fonction <
      ↪sqrt> uniquement
2 from math import * # permet l'accès à toute les fonctions
      ↪du module <math>
3 from sympy import sqrt # permet l'accès à la fonction <
      ↪sqrt> de <sympy>
4         # comme l'import de <sqrt> arrive après celle du
      ↪module <math>
5         # c'est cette dernière qui sera utilisée !
```

```
1 import math as m # les fonctions du module <math> devront
      ↪ être
2     # précédées de <m.> : exemple <m.sqrt>
3 import sympy as sy # les fonctions du module <sympy>
      ↪devront être
4     # précédées de <sy.> : exemple <sy.sqrt>
5
6 # m.sqrt(3) : fonction <sqrt> du module <math>
7 # sy.sqrt(3) : fonction <sqrt> du module <sympy>
8 # essayer les commandes suivantes
```

Quelques remarques en vrac

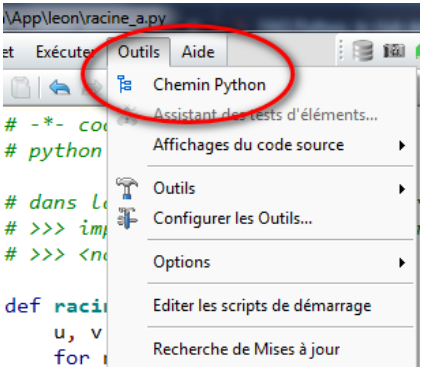
```
9  sqrt(4)
10 m.sqrt(4)
11 m.sqrt(-4)
12 sy.sqrt(-4)
```

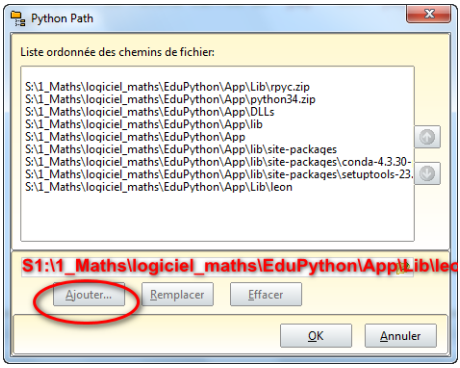
Pour connaître le contenu d'un module : `help(nom_du_module)` à vérifier pour PyScripter (`help` en console, puis une boîte de dialogue d'ouvre…)

pour importer ses propres modules dans EduPython :

Sauver le fichier dans un dossier et préciser à Python qu'il doit chercher aussi dans ce dossier.

Ici le programme s'appelle racine_a.py et il contient la fonction racine(⟨n⟩). Il est sauvé dans le dossier `EduPython\App\Leon`





On peut ensuite importer le module et utiliser la fonction racine(⟨n⟩) dans la console.

```
>>> import racine_a as fl
>>> fl.racine(45)
une approximation de racine carrée
>>>
```

pour les n° de ligne dans PyScripter : clic droit dans l'éditeur.

Quelques remarques en vrac

Variables

Python ne protège que peu de mots clés : tous les autres peuvent être redéfinis sans avertissement !

par exemple si on tape `pi=2`, alors le nombre `pi` n'est plus affecté à 3.14157… mais à 2 ! !

```
1 import keyword as kw
2 print(kw.kwlist)
3 len(kw.kwlist) # il y a 33 mots protégés en Python3
```

Les variables sont locales à la fonction, si on utilise une variable globale il faut le préciser.