Quelques remarques en vrac

Ce que je pense ;-)

⇒ 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 !

Documents

- 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*

Importance du code

Le langage Python (en hommage aux Monty Python) a été créé par Guido van Rossum. Le langage évolue grâce à sa communauté : chaque proposition d'amélioration est publique et est publiée sur le site de Python, ce sont les PEP (Python Enhancement Proposal : proposition d'amélioration de Python). Elles ne sont pas toutes acceptées, mais elles sont toutes étudiées. *https://www.python.org/dev/peps*

Recommandations de la PEP 8 :

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

Voici quelques recommandations :

Quelques remarques en vrac

- Une ligne doit contenir 80 caractères maximum.
- L'indentation doit être de 4 espaces.
- Chaque fonction est séparée de la suivante par une ligne vide.
- Les noms (variable, fonction, ...) sont composés uniquement de lettres non accentuées ou de chiffres.
  - les noms de fonction et de variables utilisent les minuscules_et_le_tiret_bas
  - les noms de constantes utilisent les MAJUSCULES_ET_LE_TIRET_BAS

Pour conclure, le poème « Zen of Python » de Tim Peters résume cette *philosophie pythonique*. Taper dans la console import this

Modules

Modules, Librairies, Bibliothèques

pour les importer :

```
from math import sqrt # dans le module <math>,
    # permet l'accès à la fonction <sqrt> uniquement
from math import * # permet l'accès à toute les fonctions
    # du module <math>
from sympy import sqrt # permet l'accès à la fonction <sqrt>
    # de <sympy>. Attention : quand deux fonctions ont le
    # même nom, c'est dernière chargée qui sera utilisée !
```

```
import math as m # les fonctions du module <math> devront
    # être précédées de <m.>
import sympy as sy # les fonctions du module <sympy>
    # devront être précédées de <sy.>

sqrt(48) #fonction <sqrt> (dernière chargée)
m.sqrt(48) #fonction <sqrt> du module <math>
m.sqrt(-48) #fonction <sqrt> du module <math>
sy.sqrt(-48) #fonction <sqrt> du module <sympy>
```

Pour connaître le contenu d'un module : help(nom_du_module) ou dir(nom_du_module)

Variables

Python ne protège que 33 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 3,141 57… mais à 2 ! !)

```
1 import keyword as kw
2 print(kw.kwlist)
3 len(kw.kwlist) # la longueur de la liste est le nb.
4     # de mots protégés en Python3
```

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