4 – Projet S4

Objectif global : Réaliser une application de gestion de notes pour un groupe d'étudiants (en mode « texte »).

4.1 – « Organisation »

- Le projet se fera par groupe 3 ou 4.
- Vous pourrez programmer et organiser vos fonctions dans plusieurs fichiers différents et les utiliser comme des bibliothèques avec les imports adéquats (voir annexe).
- Chaque fichier rendu devra contenir les tests effectués pour vérifier que les fonctions du fichier font ce qui est attendu.
- Votre programme sera exécuté de manière indépendante en mode « console » (voir ci–dessous).

Remarque

Comment lancer un programme « `mon_programme` » de manière autonome dans une console ?

1. Sous Windows : Un double-clic sur votre fichier « `python mon_programme.py` » devrait suffire ...
2. Sous Linux/Mac, ouvrir une fenêtre en mode « console » :
   (a) rechercher et lancer l'application « `terminal` ».
   (b) Taper la commande : « `python mon_programme.py` ».

Note : Pour éviter de naviguer dans vos répertoires et d'utiliser d'autres commandes, votre programme devra être placé directement dans le répertoire principal de votre compte.

4.2 – Fonctionnement global de l'application (cahier des charges)

Votre programme proposera un menu principal (voir la section « Exemples ») permettant de choisir entre les fonctionnalités suivantes :

- **Lecture** : Charge un fichier texte « notes » (avec une liste d'étudiants et leurs notes) en mémoire.
- **Affichage de la liste des étudiants avec leurs moyennes** : Affiche simplement sous forme d'un tableau la liste des étudiants triées par ordre alphabétique avec leurs notes et moyennes. La dernière ligne donnera les moyennes de la promotion.
- **Création et affichage d'une image avec le profil d'un étudiant** : Un exemple du graphique attendu (diagramme en étoile, ou diagramme radar) est donné ci–dessous (section « Exemples »).
- **Ajout d'un étudiant** : Permet de saisir un nouvel étudiant et de saisir ses notes.
- **Saisie ou modification d'une note** : Permet de saisir la note d'un étudiant pour une matière donnée. L'utilisateur pourra choisir l'étudiant et la matière. Si une note est déjà attribuée, elle sera remplacée par la nouvelle valeur.
- **Suppression d'un étudiant**
- **Suppression de tous les étudiants**
- **Sauvegarde des données** : Enregistre les données dans le fichier texte « notes ».
- **Quitter** : Quitte l'application.

Après chaque « action », le programme reviendra au menu principal.

Pour les notes, vous n'utiliserez que 5 matières (Mathématiques, Physique, Chimie, Biologie, SHS) et vous fixerez les coefficients à 1 pour chacune des matières.

Remarque

La gestion des données par l'intermédiaire d'un fichier texte que l'on écrase entièrement pour le mettre à jour (proposée ici) n'est évidemment pas une bonne façon de procéder. En pratique, on utilise pour ce genre d'application une base de données (thème qui n'est pas au programme).

4.3 – Exemples et indications

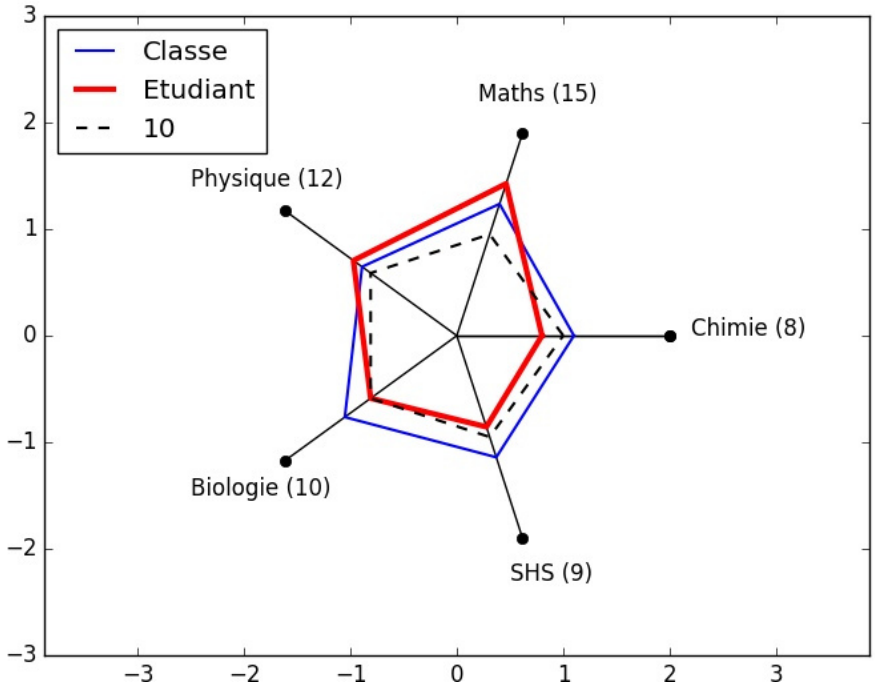
1. Un menu possible :

```
Application de gestion de notes

   (1) Charger en mémoire le fichier de notes.
   (2) Afficher la promotion.
   (3) Afficher les résultats d'un étudiant.
   (4) Ajouter un étudiant.
   (5) Saisie ou modification d'une note pour un étudiant.
   (6) Suppression d'un étudiant.
   (7) Suppression de tous les étudiants.
   (8) Sauvegarder les données.
   (9) Quitter l'application.

Donner le votre choix (un chiffre) et valider avec entrée :
```

2. Exemple de diagramme « radar » :



Pour afficher du texte à partir du point de coordonnées (a,b), on peut écrire (si on a utilisé l'alias « `import matplotlib.pyplot as plt` » en début de programme) :

```
plt.annotate('du texte', xy=(a,b))
```

3. Pour effacer l'écran dans la « console » :

   (a) Importer les commandes systèmes : `import os`

(b) Utiliser l'instruction système permettant d'effacer l'écran :

- Sous Linux ou Mac, « `clear` » : `os.system('clear')`.
- Sous Windows, « `cls` » : `os.system('cls')`.

Voir aussi l'exemple de menu disponible sous moodle.

4. Pour différencier une instruction suivant le système :

Afin que votre programme fonctionne sous plusieurs configurations sans avoir à modifier le code, on peut lui demander d'exécuter « `os.system('clear')` » si le système est de type Linux ou Mac ou « `os.system('cls')` » si le système est de type Windows.

Pour différencier le système d'exploitation (Windows ou autre par exemple) :

(a) Importer la commande « `platform` » : `from sys import platform as sys_exploitation`

(b) Comparer « `sys_exploitation` » aux noms des systèmes d'exploitation utilisés : par exemple `"win32"` ou `"win64"` pour Windows.

Voir aussi l'exemple de menu disponible sous moodle.

4.4 – Le type dictionnaire

Pour la gestion des étudiants et leurs notes, vous pourrez utiliser le type de données « dictionnaire ».

Un dictionnaire peut être vu comme une liste mais indexée par des objets immuables quelconques à la place des entiers de 0 à n. Un dictionnaire s'écrit comme une liste, entre accolades, d'éléments de type « `<cle>:<valeur>` » séparés par des virgules. Les clés seront les index et doivent donc être immuables. Un dictionnaire est lui, comme les listes, mutable. (Attention à ne pas confondre avec les ensembles.)

Le dictionnaire vide s'écrit `{}`. (Rappel l'ensemble vide s'écrit `set()`.)

Attention

Comme pour les listes, les dictionnaires sont des objets **mutables**.

Exemple 4.1.

Exemple d'utilisation d'un dictionnaire :

```
>>> annuaire = {'bob': 25, 'alice': 20, 'marc': 28}
>>> annuaire
>>> annuaire = dict([('bob',25),('alice', 20),('marc', 28)])
>>> annuaire
>>> annuaire = dict(bob=25,alice=20,marc=28)
>>> annuaire['bob'] # L'accès aux données est similaire aux listes
>>> annuaire['bob']=30 # Un dictionnaire est mutable
>>> annuaire
>>> del annuaire['marc']
>>> annuaire
>>> annuaire['kevin']=19 # Permet l'ajout d'un couple <cle;valeur> au dictionnaire
>>> annuaire
>>> 'alice'in annuaire
>>> annuaire.items() # Liste des couples
>>> annuaire.keys() # Liste des clés
>>> annuaire.values() # Liste des valeurs
>>> 'marc'in annuaire
>>> 'kevin'in annuaire
>>> len(annuaire)
```

4.5 – Fonctionnalités facultatives

- **Lecture** : L'utilisateur pourra choisir le nom du fichier à charger.
- **Sauvegarde des données** : L'utilisateur pourra choisir le nom du fichier dans lequel les données seront sauvegardées. Si le nom existe déjà on pourra afficher un avertissement et demander confirmation (le fichier précédent sera alors écrasé).
- **Saisie d'une matière** : Attention à la gestion des étudiants déjà saisis.
- **Suppression d'une matière** : Attention à la gestion des étudiants déjà saisis.
- **Modification du coefficient d'une matière** : Attention, cela va très probablement modifier la fonction de calcul d'une moyenne.

Pour les nouvelles fonctionnalités (les trois derniers items), on utilisera un fichier texte pour enregistrer les matières et leurs coefficients. Attention ! Cet ajout va entrainer **plusieurs modifications importantes de votre programme**.

Gestion des erreurs

Il faudra penser à la gestion des erreurs pour la lecture : si l'utilisateur donne un nom de fichier à lire qui n'existe pas ... On demandera une nouvelle saisie par exemple.

ANNEXES

A – Modules

Il est possible de créer ses propres modules regroupant des fonctions spécifiques et de les utiliser via un « import » comme d'autres modules existant (`math`, `random`, ...).

Il suffit pour cela, après avoir sauvegarder son fichier servant de module, de l'importer avec la commande `import` et le nom du fichier.

Exemple 1.1.

Télécharger sur moodle les deux fichiers suivants et les enregistrer dans un même dossier.
Puis exécuter `autre_module.py`.

- `mon_module.py`

```
# -*- coding:utf-8 -*-
def sommeNpremiersEntiers(n):
    """calcule la somme des n premiers entiers"""
    s=0
    for i in range(1,n+1):
        s+=i
    return s

N=int(raw_input("Donnez un entier : "))
somme=sommeNpremiersEntiers(N)
print "1 + ... +",N,"=",somme
print "Verification :"
print "n(n+1)/2 =",N*(N+1)/2.
```

- `autre_module.py`

```
# -*- coding:utf-8 -*-
import mon_module as mm

for i in range(10,12):
    S=mm.sommeNpremiersEntiers(i)
    print "1 + ... +",i,"=",S
```

Pour éviter le problème d'exécution de code non désirée lors de l'import d'un module, on peut utiliser l'instruction « `if __name__ == "__main__":` » qui spécifie à Python de n'exécuter ce qui suit qui si le module dans lequel on se trouve est le programme principal (donc pas un « import »).

Exemple 1.2.

Il faudrait modifier le programe `mon_module.py` en ajoutant « `if __name__ == "__main__":` » de la façon suivante :

```
⋮
if __name__ == "__main__":
    N=int(raw_input("Donnez un entier : "))
    somme=sommeNpremiersEntiers(N)
⋮
```

B – Bibliographie

[1] B. WACK, *Informatique pour tous*, Éditions Eyrolles, 2013

[2] E. LE NAGARD, *Informatique, Initiation à l'algorithmique en Scilab et Python*, Éditions Pearson, 2013

[3] A. CASAMAYOU-BOUCAU, P. CHAUVIN, G. CONAN, *Programmation en Python pour les mathématiques*, Éditions Dunod, 2012

[4] G. SWINNEN, *Apprendre à Programmer avec Python*, Éditions O'Reilly, 2004

[5] S. CHAZALLET, *Python, Les fondamentaux du langage*, Éditions ENI, 2012