Examen HMPH104 – Modélisation et Simulation en Physique – Session 1 – 2015-2016
Sujet de David Cassagne (durée 2h) – Mardi 12 janvier 2016
(Tous documents autorisés)

Vous nommerez vos fichiers : "exercice1a.py", "exercice1b.py", "exercice2a.py", etc.
Les premières lignes des fichiers seront :

```
# Nom : VotreNom
# Prenom : VotrePrenom
# Exercice 1a
```

Exercice 1 : Visualisation

a) Visualisation 2D : Ecrire un programme qui permet de visualiser la fonction $f(x,y)=\frac{\sin(2x)\sin(3y)}{6xy}\ln\left(4+x^2+y^2\right)$ sur l'intervalle $[-\pi,\pi]\times[-2\pi,2\pi]$ avec un maillage de 71×141 points suivant l'axe des *x* et des *y* respectivement.

b) Visualisation d'une fonction à valeurs complexes : Ecrire un programme pour représenter la fonction $f(x)=\frac{1}{1+(x-1)^2}e^{-i\pi x}$ sur l'intervalle $[-3,4]$ avec 71 valeurs de *x* en utilisant une échelle colorée pour visualiser les valeurs de l'argument.

Exercice 2 :

a) Créer une classe *Fruits* permettant de gérer des fruits en prenant en compte leur prix par kg en euros et leur poids en kg. Cette classe possédera 2 attributs **privés** : *__prix_par_kg* et *__poids*. Cette classe disposera des méthodes suivantes :
- un constructeur qui permet d'initialiser le prix par kg et le poids des fruits.
- une méthode **`prix()`** qui renvoie le prix des fruits en tenant compte de leur poids.

Créer un programme principal qui définit une instance de 1.2 kg de fruits à 2.3 euros/kg et affiche le prix.

b) Définir une classe *Pommes* **dérivée** de la classe *Fruits*. Cette classe disposera des mêmes fonctionnalités que la classe *Fruits* mais elle possèdera en plus un attribut *variete* qui est une chaine de caractères qui contient la variété des pommes. On définira un constructeur qui permettra d'initialiser la variété, le prix par kg et le poids. On définira une méthode **`affiche()`** qui affichera `Pommes`, le nom de la variété et le prix en tenant compte du poids, exemple :

```
Pommes variete golden 3.6 euros
```

Ajouter un programme principal qui donne un exemple d'utilisation de cette classe.

c) Définir une classe *Raisins* **dérivée** de la classe *Fruits*. Cette classe disposera des mêmes fonctionnalités que la classe *Fruits* mais elle possèdera en plus un attribut *pepins* qui est un booléen qui caractérise si les raisins sont avec ou sans pépins. On définira un constructeur qui permettra d'initialiser la présence ou non de pépins, le prix par kg et le poids. On définira une méthode **`affiche()`** qui affichera `Raisins` avec ou sans pépins et le prix , exemple :

```
Raisins avec pepins 2.8 euros
```

Ajouter un programme principal qui donne un exemple d'utilisation de cette classe.

d) Modifier à présent la classe *Fruits* de la question (a) pour lui ajouter une méthode de classe **`poids_total()`** qui renvoie le poids total de l'ensemble des fruits qui ont été instanciés. Ajouter un programme principal qui, en utilisant les classes *Pommes* et *Raisins* des questions (b) et (c), crée des instances de ces deux fruits et affiche ensuite le poids total.

Exercice 3 : Interface graphique – climatisation réversible

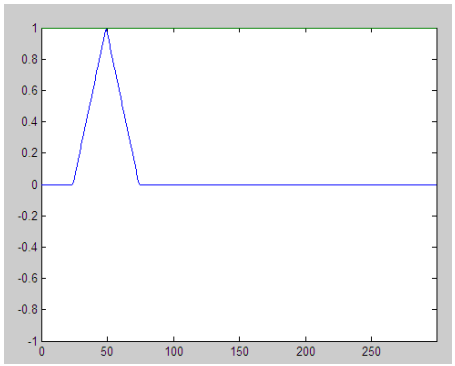
Créer une interface graphique permettant de simuler le fonctionnement du programmateur d'une climatisation réversible dont la température sera affichée avec des valeurs entières allant de -10 à 50.

a) Dans un premier temps, la fenêtre comprendra deux zones d'affichage : une pour la température et une qui affichera « Mode Chauffage » ou « Mode Climatisation » ou rien quand l'appareil est à l'arrêt. Elle disposera également de deux cases à cocher : « Marche/Arrêt » et « Été/Hiver ». La case « Marche/Arrêt » permettra d'allumer ou d'éteindre l'appareil. Quand la case sera cochée, cela correspondra à l'état de marche et la température affichée la première fois sera fixée à 20. La case « Été/Hiver » permettra de passer du « Mode Chauffage » au « Mode Climatisation ». Quand elle sera cochée cela correspondra à l'été et on affichera « Mode Climatisation ». Quand elle sera décochée cela correspondra à l'hiver et on affichera « Mode Chauffage ». Quand la case « Marche/Arrêt » sera décochée, cela correspondra à l'état d'arrêt et aucune valeur ne sera affichée dans les zones d'affichage. Si on recoche la case à nouveau, les anciennes informations affichées doivent réapparaitre.

b) Ajouter deux boutons « + » et « - ». Ces boutons permettront d'augmenter ou de diminuer la puissance en restant dans le domaine précisé au début. Ces boutons ne seront pas actifs quand l'appareil est à l'arrêt.

Exercice 4 : Animation – Paquet d'ondes triangulaire

Ecrire un programme qui visualise sous forme d'animation la propagation vers la droite à vitesse constante et sans déformation d'un paquet d'ondes de forme triangulaire (voir ci-dessous).



Exercice 5 : Animation – Corpuscule dans une boite

Ecrire un programme qui visualise le déplacement 2D d'un corpuscule ponctuel dans une boite rectangulaire. Le corpuscule sera situé au départ au centre de la boite. On choisira une vitesse arbitraire. La direction de déplacement sera issue d'un tirage aléatoire avec équiprobabilité entre toutes les directions de l'espace de simulation. On considérera que le corpuscule n'est soumis à aucune force et suit donc un mouvement rectiligne uniforme mais on gérera les rebonds sur les parois de la boite.

Exercice 6 : Interface graphique - Echanges de messages entre deux fenêtres

Créer une interface graphique avec deux fenêtres permettant de réaliser des échanges de messages entre Alice et Bob. Une fenêtre sera nommée « Fenetre Alice », et l'autre « Fenetre Bob ». Chaque fenêtre contiendra un champ de texte pour saisir un message, un bouton d'envoi et une zone pour afficher un message reçu. Quand Alice aura saisi un message dans son champ de texte, il sera affiché dans la fenêtre de Bob lorsqu'elle appuiera sur envoi. Un comportement réciproque se produira pour un message saisi dans la fenêtre de Bob.