Université Paul Sabatier Novembre 2015

F.S.I **TP N°2** CB

Exercice 1. *Fabriquer une table à partir d'une formule*

Écrire un programme qui affiche une table correctement formattée de t et $y(t)$ où

$$y(t) = v_0 t - \frac{1}{2} g t^2.$$

Utiliser n valeurs de t uniformément réparties dans l'intervalle $[0, 2v_0/g]$. Prendre $v_0 = 1$ et $n = 11$.

Répéter l'exercice de telle manière que les valeurs de t et de y soient stockées dans deux listes `t` et `y`. Ensuite, parcourir les listes avec une boucle `for` et afficher les valeurs des valeurs des tables t et y.

Exercice 2. *Travailler avec une liste*

Positionner une variable `premier` a une liste contenant les nombres 2, 3, 5, 7, 11 et 13. Afficher chaque élément de la liste à l'aide d'une boucle `for`. Assigner 17 à la variable `p` et ajouter `p` à la fin de la liste. Afficher l'intégralité de la nouvelle liste.

Exercice 3. *Générer des coordonnées équiréparties*

On veut générer x coordonnées entre 1 et 2 avec un écart de 0.01. Les coordonnées sont données par la formule $x_i = 1 + ih$, où $h = 0.01$ et i parcourt les entiers $0, 1, \cdots, 100$. Calculer les valeurs x_i et les stocker dans une liste (utiliser une boucle `for`, et ajouter chaque nouvelle valeur x_i à la liste, qui est initialement vide).

Exercice 4.

On considère la liste suivant `q=[['a','b','c'],['d','e','f'],['g','h']]`. Donner les instructions pour extraire

1. la lettre `a`
2. la liste `['d','e','f']`
3. le dernier élément `h`.

On peut visiter tous les éléments de `q` avec le programme suivant

```
for i in q:
    for j in range(len(i)):
        print i[j]
```

De quel type d'objets sont `i` et `j` ?

Exercice 5. *Écrire une fonction de conversion Fahrenheit-Celsius*

La formule pour convertie des degrés Fahrenheit vers des Celsius s'écrit

$$C = \frac{5}{9}(F - 32).$$

Écrire une fonction `C(F)` qui met en oeuvre cette formule.

Exercice 6. *Écrire une fonction pour résoudre $ax^2 + bx + c = 0$.*

Écrire une fonction `root(a,b,c)` qui renvoie les deux racines de l'équation. Le résultat renvoyé par `root` devra être des objets réels quand les racines le sont, et des objets complexes sinon.

Exercice 7.

La fonction standard Python appelée `sum` prend une liste comme argument et calcule la somme des éléments dans cette liste :

```
>>> sum([1,3,5,-5])
4
```

Écrire votre propre version de `sum`.

Exercice 8. *Intégration par la formule des trapèzes.*

Une approximation de l'intégrale de la fonction f sur un intervalle $[a, b]$ peut être trouvée en commençant par approcher $f(x)$ par une ligne droite qui s'étend entre les points extrémaux $(a, f(a))$ et $(b, f(b))$, et ensuite en calculant l'aire sous la ligne droite (qui est l'aire d'un trapèze). La formule résultante est

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}(f(a)+f(b)). \tag{1}$$

Écrire une fonction `trapezint1(f,a,b)` qui renvoie l'approximation de l'intégrale. L'argument `f` est une implémentation Python de la fonction mathématique f.

Utiliser cette fonction pour calculer les valeurs approchées des intégrales $\int_0^{\ln 3} e^x\,dx$, $\int_0^{\pi} \cos x\,dx$, $\int_0^{\pi} \sin x\,dx$ et $\int_0^{\pi/2} \sin x\,dx$. Dans chaque cas, afficher l'erreur, c'est à dire la différence entre l'intégrale exacte et l'approximation.

Exercice 9. *Intégrer une fonction par deux trapèzes.*

On peut facilement améliorer l'approximation de l'exercice précédent en approchant l'aire sous la courbe $y = f(x)$ par deux trapèzes de même largeur. Écrire la formule pour cette approximation et la mettre en oeuvre dans une fonction `trapezint2(f,a,b)`. Tester cette fonction sur les exemples de l'exercice précédent et comparer la qualité de l'approximation.

Exercice 10. *La règle générale de l'intégration par la formule des trapèzes.*

On a vu dans l'exercice précédent qu'il est possible d'améliorer le calcul approché d'une intégrale en divisant l'aire sous la courbe $y = f(x)$ en deux trapèzes. L'idée est de généraliser en considérant maintenant n trapèzes de largeur identique. Montrer que la formule d'approximation est alors

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} \frac{1}{2} h\left(f(x_{i-1}) + f(x_i)\right),$$

où h est la largeur des trapèzes $h = (b-a)/n$ et $x_i = a+ih$, $i = 0, \cdots, n$, sont les coordonnées des côtés des trapèzes. Mettre en oeuvre une fonction Python `trapezint(f,a,b,n)`. L'essayer sur les exemples des exercices précédents pour $n = 10$ puis comparer les résultats en termes d'erreur.

Exercice 11. *Calcul de la longueur d'un chemin.*

Considérons un objet ponctuel qui se déplace dans le plan (O, x, y). Suivant n points en temps, on enregistre les positions respectives (x, y) de cet objet : (x_0, y_0), (x_1, y_1), $\cdots$, (x_{n-1}, y_{n-1}). La longueur totale L du chemin de (x_0, y_0) à (x_{n-1}, y_{n-1}) est la somme des longueurs des segments individuels $((x_{i-1}, y_{i-1})$ à (x_i, y_i), $i = 1, \cdots, n-1$:

$$L = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}.$$

Fabriquer une fonction `longueurchemin(x,y)` pour calculer L qui vérifie cette formule. Les arguments `x` et `y` contiennent $x_0, \cdots, x_{n-1}$ et $y_0, \cdots, y_{n-1}$, les coordonnées respectives. Tester la fonction sur un chemin triangulaire de quatre points $(1, 1)$, $(2, 1)$, $(1, 2)$ et $(1, 1)$.

Exercice 12. *Approximation de π*

La valeur de π est égale à la circonférence d'un cerle de rayon 1/2. On suppose qu'on approche le cercle par un polygone régulier avec $N+1$ points. La longueur de ce polygone (son périmètre) peut être calculé à partir de la fonction `longueurchemin(x,y)` de l'exercice précédent. Calculer $N+1$ points (x_i, y_i) le long du cercle de rayon 1/2 avec les formules

$$x_i = \frac{1}{2}\cos(2\pi i/N), \quad y_i = \frac{1}{2}\sin(2\pi i/N), \quad i = 0, \cdots, N.$$

Appeler la fonction `longueurchemin` et écrire l'erreur d'approximation de π pour $N = 2^k$, $k = 2, 3, \cdots, 10$.

Exercice 13. *Approximation d'une fonction par une somme de sinus.*

On considère la fonction constante par morceaux

$$f(t) = \begin{cases} 1, & 0 < t < T/2, \\ 0, & t = T/2, \\ -1, & T/2 < t < T. \end{cases}$$

On peut approcher f par la somme

$$S(t;n) = \frac{4}{\pi}\sum_{i=1}^{n} \frac{1}{2i-1}\sin\left(\frac{2(2i-1)\pi t}{T}\right).$$

Il peut ête montré que $S(t,n) \to f(t)$ quand $n \to \infty$.

Écrire une fonction Python `S(t,n,T)` qui renvoie la valeur de $S(t;n)$. Écrire aussi une fonction Python `f(t,T)` pour calculer $f(t)$. Afficher les erreurs $f(t) - S(t;n)$ pour différentes valeurs de n et de t pour les cas où 1, 3, 5, 10, 30, 100 et $t = \alpha T$, avec $T = 2\pi$, et $\alpha = 0.01$, 0.25, 0.49.

Remarque. Une somme de fonctions sinus et/ou cosinus est appelée une *série de Fourier*. Approcher une fonction par une série de Fourier est une technique très importante en science et en technologie.

Exercice 14. *Écrire une fonction de différentiation numérique*

La formule

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

peut être utilisée pour calculer une valeur approchée de la dérivée d'une fonction mathématique f quand h est petit. Écrire une fonction `diff(f,x,h)` qui renvoie une approximation de la dérivée d'une fonction mathématique représentée par une fonction Python `f(x)`.

Appliquer la fonction `diff` pour différentier $f(x) = e^x$ en $x = 0$, $f(x) = e^{-2x^2}$ en $x = 0$, $f(x) = \cos x$ en $x = 2\pi$ et $f(x) = \ln x$ en $x = 1$. Utiliser $h = 0.01$. Dans chaque cas, afficher l'erreur commise entre la dérivée approchée et la dérivée exacte pour différentes valeurs de x.

Faire la même chose pour la formule d'approximation de la dérivée seconde

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

Écrire une fonction `diff2(f,x,h)` et procéder aux mêmes tests.

Exercice 15. *Fabriquer une approximation numérique d'une intégrale par une méthode des trapèzes adaptative.*

Un problème avec la formule des trapèzes (**??**) est de décider combien de trapèzes (n) il faut utiliser afin d'atteindre une précision désirée. Soit E l'erreur dans la méthode des trapèzes, i.e., la différence entre l'intégrale exacte et celle produite par la formule des trapèzes. On aimerait prescrire une (petite) tolérance ε et trouver un n tel que $E \leq \varepsilon$. Ceci demande une expression de l'erreur E en fonction de n

On peut montrer que

$$E \leq \frac{1}{12}(b-a)h^2 \max_{x\in[a,b]} \left|f''(x)\right|. \tag{2}$$

Le maximum de $|f''(x)|$ peut être calculé de manière approchée en évaluant f'' en un grand nombre de points dans $[a, b]$, en prenant la valeur absolue $f''(x)$, et en trouvant la valeur maximale. On peut utiliser pour cela la fonction `diff2` de l'exercice précédent.

Avec l'estimation calculée de $\max |f''(x)|$, on peut trouver h en calculant la pire erreur dans (**??**) et en l'égalisant à la tolérance désirée

$$\frac{1}{12}(b-a)h^2 \max_{x\in[a,b]} \left|f''(x)\right| = \varepsilon.$$

Après calculs, on trouve h donné par

$$h = \sqrt{12\varepsilon} \left((b-a) \max_{x\in[a,b]} \left|f''(x)\right| \right)^{-1/2}.$$

Avec $n = (b-a)/h$, on a le n qui correspond à la précision désirée ε.

Fabriquer une fonction Python `trapezeint_adapt(f,a,b,eps)` pour calculer l'intégrale $\int_a^b f(x)\,dx$ avec une erreur inférieure ou égale à ε. On calculera d'abord n puis on appelera `trapezint(f,a,b,n)`.

Exercice 16. *Fabriquer une table d'approximation de* $\cos x$

La fonction cos peut être approchée par la somme

$$C(x;n) = \sum_{j=0}^{n} c_j,$$

où

$$c_j = -c_{j-1}\frac{x^2}{2j(2j-1)}, \quad j = 1, 2, \cdots, n,$$

et $c_0 = 1$. Fabriquer une fonction Python pour calculer $C(x; n)$. (Astuce : représenter c_j par une variable `terme`, et la mettre à jour par `term = -term*...` dans une boule `for`, et accumuler la variable `term` dans une variable pour la somme.)

Fabriquer aussi une fonction qui affichera la table des erreurs d'approximation entre $C(x; n)$ et $\cos(x)$ pour différentes valeurs de x et de n qui seront données en arguments de la fonction. Les valeurs de x parcoureront la première colonne et celles de n seront parcourues en ligne. Par exemple, la table pour $x = 4\pi$, 6π, 8π, 10π et $n = 5$, 25, 50, 100, 200 pourra ressembler à

```
   x          5         25         50         100        200
12.5664   1.61e+04  1.87e-11   1.74e-12   1.74e-12   1.74e-12
18.8496   1.22e+06  2.28e-02   7.12e-11   7.12e-11   7.12e-11
25.1327   2.41e+07  6.58e+04  -4.87e-07  -4.87e-07  -4.87e-07
31.4159   2.36e+08  6.52e+09   1.65e-04   1.65e-04   1.65e-04
```