INFO-H-100 - Programmation
TP 15 - Pylab

Ex. 1. `pylab` est un module python qui regroupe des librairies de calcul scientifique dont `matplotlib`, une librairie permettant d'afficher des graphiques (courbes, histogrammes, ...).

Ecrivez le code suivant dans un fichier et exécutez le avec Python 2.7.

```
import pylab
import numpy

x = numpy.linspace(-10, 10, 1000)
y = map( lambda x: x**2, x )

pylab.plot(x,y)
pylab.show()
```

Ex. 2. A l'aide de `matplotlib`, et de la fonction précédente, écrire une fonction qui reçoit en paramètres une fonction f, une borne inférieure beg, une borne supérieure end, et un nombre de termes n, et qui trace la fonction f de beg à end.

```
>>> import math
>>> plot(math.sin, -1, 1, 100)
```

Ex. 3. Dans la documentation de matplotlib, trouvez comment choisir la couleur de chaque courbe ainsi que comment associer du texte aux axes. Testez ces fonctionnalités. Conseil : lisez les exemples sur

- `http://matplotlib.sourceforge.net/users/pyplot_tutorial.html`
- `http://matplotlib.sourceforge.net/api/pyplot_api.html#matplotlib.pyplot.plot`

Ex. 4. Ecrire une fonction qui dessine sur le même graphique les fonctions $f(x)$ passées en paramètres via une liste de tuples (fonctions, couleur) (plusieurs `plot` avec un `show` en toute fin). La fonction devra également prendre en paramètres les points à calculer, toujours au format attendu par `linspace`.

```
>>> import math
>>> multi_plot([(math.sin, "r"), (lambda x:math.cos(0.5*x), "g")], -5, 5, 1000)
```

Dessinez les fonctions $sin(x)$, $2sin(x)$, $sin(2x)$ et $sin(x^2)$ pour tous les 10000 réels x entre 0 et 10.

Ex. 5. Dans un travail d'ingénieur, il arrive souvent de devoir construire un modèle mathématique sur base d'un ensemble de mesures prises. Vous trouverez dans le fichier `data_tp_15.txt` un ensemble de données simulant des mesures prises en effectuant des expériences. Chaque ligne représente une expérience ; lors de chaque expérience, deux mesures sont prises. On veut construire un modèle permettant de prédire la seconde mesure en fonction de la première.

Dans ce genre de situation, une bonne première idée est toujours de visualiser les données sous la forme d'un nuage de points (voir la documentation de `pylab` pour afficher les points sous la forme de points plutôt que de courbe). Pour cet exercice, affichez ce nuage de points en utilisant `pylab`.

Pour lire le fichier, vous pouvez utiliser le module fourni `tp15_read_data` et sa fonction `read`. Consultez l'aide de cette fonction pour voir comment l'utiliser.

Ex. 6. Dans le module `numpy` se trouve une fonction appelée `polyfit` qui effectue l'ajustement d'un polynôme sur un ensemble de points. Son interface est la suivante : dans sa version la plus simple, elle prend comme arguments un vecteur de x, un vecteur de y, et un degré souhaité, et renvoie un polynôme du degré demandé. Par exemple :

```
>>> import numpy
>>> numpy.polyfit([1, 2, 3], [2, 4, 6], 1)
array([  2.00000000e+00,   4.98488902e-15])
```

Deux remarques importantes :

– Le polynôme renvoyé est dans le sens inverse de la représentation utilisée dans le syllabus
– `numpy` définit un type `array`, qui est différent de la liste de base de Python. Vous pouvez néanmoins accéder aux éléments d'un `array` de la même manière qu'à ceux d'une `list` (notation indicée et slices). Vous pouvez convertir un `array` en `list` à l'aide de la fonction `list` :

```
>>> list(numpy.polyfit([1, 2, 3], [2, 4, 6], 1))
[1.9999999999999989, 4.9848890172662702e-15]
```

Utilisez la méthode `numpy.polyfit` pour ajuster des polynômes de degrés 1, 2, 3, 10 et 40 sur les points du fichier `data_tp_15.txt`. Visualisez graphiquement les résultats, en traçant, pour chacun des cinq polynômes obtenus, le polynôme et le nuage de points sur un même graphe.

Ex. 7. Le fichier `tp15_salaire.txt` contient le résultat d'un sondage sur les salaires des ingénieurs entre 1 et 3 ans après leur sortie de l'université. Sur base du module fourni à l'exercice 5, importez ce jeu de données dans votre programme. Tracez ensuite un histogramme de ces données avec `pylab.hist`.

Solution de l'exercice 2:

```
import pylab
import math
from numpy import linspace

def plot( fn, bi, bs, n ):
    x = numpy.linspace(bi, bs, n)
    y = map( fn, x )
    pylab.plot(x,y)
    pylab.show()

plot( math.sin, -1, 1, 100 )
```

Solution de l'exercice 3:

Par exemple :

```
import pylab
import math
from numpy import linspace

def test_colors_and_legend():
    x = linspace( -10, 10, 1000 )
    y1 = map( math.sin, x )
    y2 = map( lambda z: 2*math.sin(2*z), x )

    pylab.plot(x, y1, 'g', label='sin(x)')
    pylab.plot(x, y2, 'r', label='2 sin(2x)')
    pylab.xlabel("x")
    pylab.ylabel("y")
    pylab.legend()
    pylab.show()

test_colors_and_legend()
```

Solution de l'exercice 4:

```
import pylab
import numpy
import math

def multi_plot( fn_col_tuple, beg, end, n ):
    x = numpy.linspace( beg, end, n )
    for (fn,col) in fn_col_tuple:
        y = map( fn, x )
        pylab.plot( x, y, col )
    pylab.show()

multi_plot([(math.sin, "r"), (lambda x: 2 * math.sin(x), "g"),
    (lambda x: math.sin(2*x), "c"), (lambda x: math.sin(x * x), "m")],
    -5, 5, 1000)
```

Solution de l'exercice 5:

```
from tp15_read_data import read
import pylab

x, y = read("data_tp_15.txt")

pylab.plot(x, y, "ro")
pylab.show()
```

Solution de l'exercice 6:

```
from tp15_read_data import read
from numpy import polyfit, polyval
import pylab

def main():
    x, y = read("data_tp_15.txt")
    polys = calc_polys(x, y, [1, 2, 3, 10, 40])
    plot_polys(polys, x, y)

def calc_polys(x, y, degs):
    res = []
    for deg in degs:
        res.append(polyfit(x, y, deg))
    return res

def plot_polys(polys, xs, ys):
    for poly in polys:
        plot_poly(poly, xs, ys)

def plot_poly(poly, xs, ys):
    yp = polyval(poly, xs)
    pylab.plot(xs, ys, "ro")
    pylab.plot(xs, yp)
    pylab.show()

if __name__ == "__main__":
    main()
```

Solution de l'exercice 7:

```
import pylab

def read_one_col(filename):
    xs = []
    with open(filename) as f:
        for line in f:
            x = float( line.strip() )
            xs.append( x )
    return xs

pylab.hist( read_one_col("tp15_salaire.txt") )
pylab.show()
```