Initiation à Python

fond$_t$ *itrefinal*

Licence 3ième année de Mathématiques

Année 2019-2020

Bibliographie

Python

- `http://www.python.org`
- `https://docs.python.org/3.6/tutorial/index.html`
- `https://docs.scipy.org/`
- A. Casamayou-Boucau, P. Chauvin, G. Connan, *Programmation en Python pour les mathématiques*, Dunod, 2016.
- B. Wack et al , *Informatique pour tous en classes préparatoires aux grandes écoles*, Eyrolles, 2013.
- Lionel Uhl, *1001 codes Python pour la modélisation*, Ellipse, 2014.

Beaucoup de documents et programmes sur le web !

Historique

- un langage créé en 1989 par Guido van Rossum
  - nom vient des Monty Python
- un langage de programmation objet, multi-paradigme et multiplateformes.
- un langage utilisé par Google, la NASA, ...
- le langage de logiciels tels que Blender, Paraview, ...
- un langage de plus en plus utilisé dans l'industrie
  - dans la communauté scientifique, avec le développement des librairies :
    - Calcul : NumPy, SciPy, PyIMSL Studio, Sympy, SAGE
    - Analyse de données : pandas
    - Visualisation : pydot, maplotlib, pyngl, ...
    - Chimie : PyMOL, MMTK, Chimera, ...
    - Biologie : Biopython
  - dans la formation (lycée, classes prépa,...)

Les distributions Python

Choisir si possible la distribution

- Anaconda `https://www.anaconda.com/download`
- ou Miniconda `https://conda.io/miniconda.html`

Autres possibilités :

- $\text{fond}_p agebis$ pour GNU/Linux Ubuntu $\geq$ 14.04 : installation standard via les dépôts officiels Ubuntu de Canonical

Version utilisée : Python 3.*

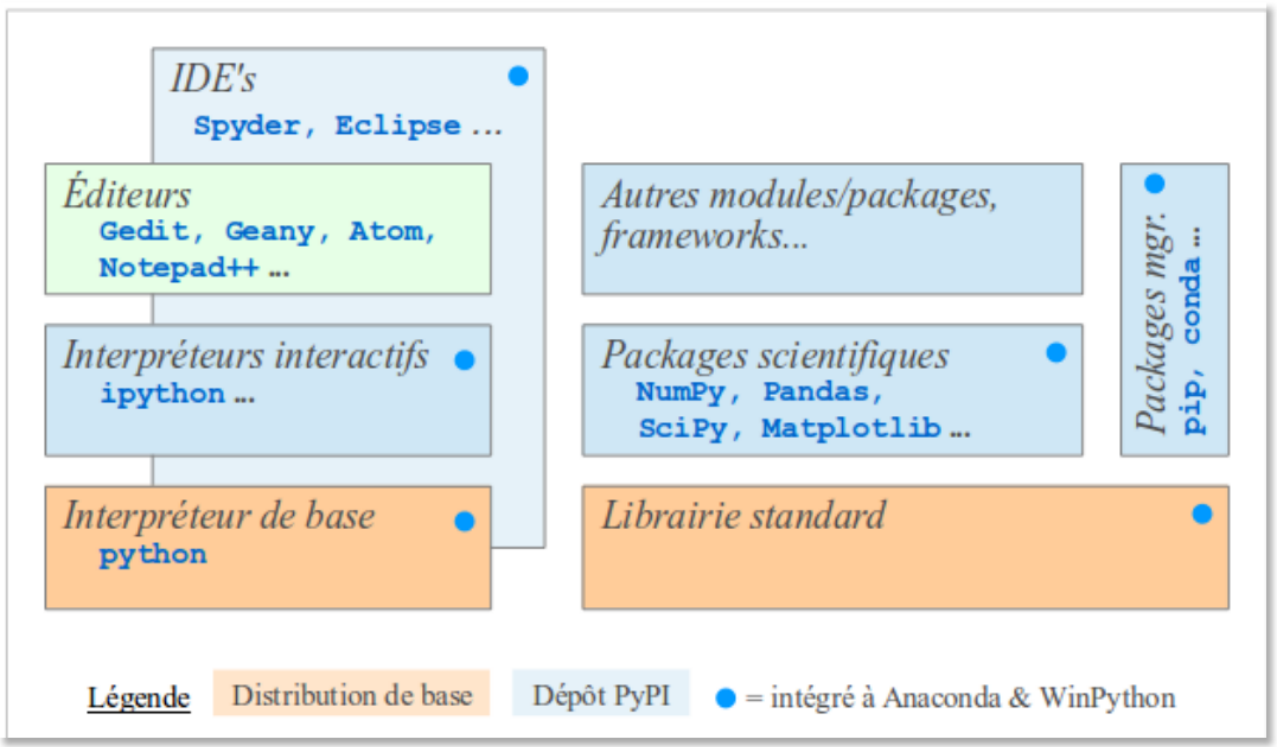
Outils de développement

Les interpréteurs les plus utilisés

- python ou cpython : interpréteurs de base
- ipython : interpréteur interactif très évolué

Les différentes méthodes pour utiliser Python

1. Dans la console (`python, ipython, ...`)
2. Dans un éditeur de texte : (gedit, emacs,...)
3. Utiliser un environnement de programmation : **Spyder**, Jupyter (notebook), Pyzo (version disponible aux concours) `http://www.pyzo.org`,...



L'environnement Spyder

fond$_p$*agebis*

L'environnement Notebook

fond$_p$*agebis*

Objectifs du stage

1. **Un exemple de code python**
2. **Python : une maxi calculatrice**
   - Les variables
   - Les fonctions
   - Les boucles et les tests
3. **Les modules Python**
   - Les librairies classiques
     - Fonctions mathématiques : math
     - Calcul scientifique : numpy, scipy,SymPy
     - Graphiques : matplotlib.pyplot

   $\text{fond}_p agebis$
4. **Pour aller plus loin**
   - Les entrées-sorties
   - Les modules personnels
   - Les fonctions
5. **Exercices pour aller plus loin**

Objectifs du stage

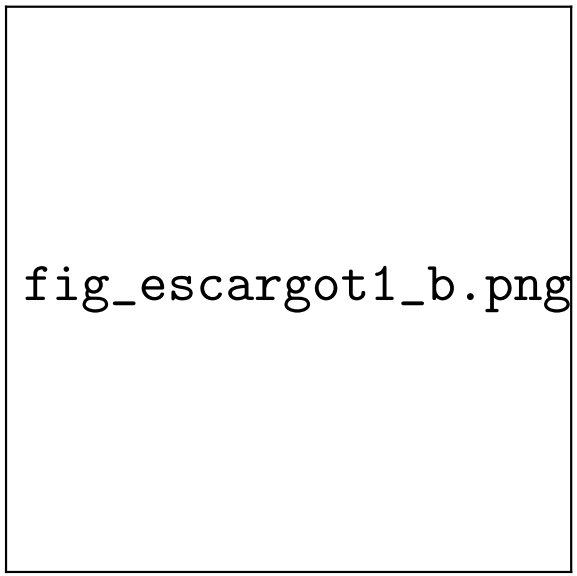
fond$_p$*agebis*

Un exemple classique : les suites récurrentes

$$u_{n+1} = f(u_n)$$

pour

$$f(x) = x^2$$




```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**2

def visualise(x0,fonction,p,q,xmin,xmax,ymin,ymax):
    l=np.size(x0)
    xx=np.linspace(xmin,xmax,100)

    plt.figure(q)
    plt.clf()
    plt.plot(xx,xx,'-.c',lw=3)
    plt.plot(xx,fonction(xx),'-.m',lw=3)

    for s in range(l):
        x=np.zeros(p+1);yy=np.zeros((2*p,2))
        x[0]=x0[s]
        for i in range(p):
            x[i+1]=fonction(x[i])
            yy[2*i,0]=x[i];yy[2*i,1]=x[i+1];yy[2*i
                +1,0]=x[i+1];yy[2*i+1,1]=x[i+1]
        plt.plot(yy[:,0],yy[:,1])
        plt.axis([xmin,xmax,ymin,ymax])

visualise(np.array([0.5,0.8,1.1]),f,5,1,  0,4,0,4)
```

Objectifs du stage

fond$_p$*agebis*

La calculatrice élémentaire

Entiers, décimaux, bouléens, chaines de caractéres

```
>>> i=10
>>> i
10
>>> x=10.0
>>> x
10.0
>>> print(x)
10.0
>>> bol=True
>>>bol
True
>>>texte='bonjour'
>>> print(texte)
bonjour
>>> texte
'bonjour'
>>>texte2=''c est lundi'' # commentaires
>>> texte2
"c est lundi"
>>> print(texte2)
c'est lundi
```

La calculatrice élémentaire

Addition, soustraction, division, puissance

```
>>> 1+1
2
>>> 7-3
4
>>> 22/3
7.333333333333333
>>> 2**4
16
```

Division entre entiers : reste,...

```
>>> int(21/4)
5
 >>> 21%4
1
```

Opérations sur les chaines de caractères

```
>>> texte1='lundi'
>>> texte2="Aout"
>>> j=28; a=2017
>>> texte=texte1+' '+str(j)+texte2+str(a)
>>> texte
'lundi 28Aout2017'
```

La calculatrice élémentaire

Les listes ou tableaux

Ce sont des collections hétérogènes, ordonnées et modifiables d'éléments, séparés par des virgules, et entourées de crochets.

```
>>> saison=['printemps','ete','automne','hiver']
>>> saison[0];saison[2] # 1er et 3eme terme de la liste saison
'printemps'
'automne'
>>>maliste=['chat',10,1.5]
>>> maliste
['chat', 10, 1.5]
>>> maliste[1]=0.2
>>> maliste
['chat', 0.2, 1.5]
>>> maliste[:2]
['chat', 0.2]
>>> maliste[0:2]
['chat', 10]
>>> maliste[0:3:2]
['chat', 1.5]
>>> maliste[-1]
>>> age=[18,19,18,20]
>>> taille=['1m70','1m60','1m75','1m79']
>>> eleve=[age,taille];eleve[1][2]
'1m75'
```

La calculatrice élémentaire

Quelques listes prédéfinies

```
>>> maliste=[]
>>> maliste
[]
>>> liste1=range(3) # classe range(n) contenant les entiers de 0 a n-1
>>> liste1
range(0, 3)
>>> list(liste1) # transforme cette suite en liste
[0, 1, 2]
>>> list(range(2,6)) # liste des entiers allant de 2 a 5
[2, 3, 4, 5]
>>> list(range(1,9,3)) # liste des entiers de 1+3k inf strict a 9
[1, 4, 7]
>>> l=[0.0] * 3 # [0.0, 0.0, 0.0]
```

La calculatrice élémentaire

Quelques opérations

```
>>> liste2=list(range(5))
>>> liste2.append(9) # insere la valeur 9 en fin de liste
>>> liste2
[0, 1, 2, 3, 4, 9]
>>> liste2[3]
3
>>> del liste2[3] # enleve la 4eme valeur de la liste
>>> liste2
[0, 1, 2, 4, 9]
>>> liste2[3:4]=['toto',10] # intercale deux termes
>>> liste2
[0, 1, 2, 'toto', 10, 9]
>>> len(liste2)
6
```

La calculatrice élémentaire

Les n-uplets

Ce sont des collections hétérogènes, ordonnées et non modifiables d'éléments, séparés par des virgules, et entourées de parenthèses.

```
>>> arguments=(10,2.0,True)
```

fond_pagebis

- Contrairement aux listes, impossible d'ajouter ou de retirer un élément
- Manipulation rapide
- Utilisés comme paramètres de fonctions comme `odeint`

La calculatrice élémentaire

Affectation des variables

```
>>> x=y=2.0 # affectation multiple
>>> x
2.0
>>> y=1
>>> x
2.0
>>> t=x<2 # definition d'un booleen
>>> t
False
>>> print('x=',x) # ecriture dans la console
x= 2.0
>>> i=2
>>> print(i==2)
True
>>> u,v=1.6,'toto' # affectation multiple
>>> u
1.6
>>> v
'toto'
>>> u+=1 # ajouter 1 a u
>>> u
2.6
>>> del(u) #  effacer la variable u
```

La calculatrice élémentaire

Type des variables

Python détecte automatiquement le type des variables !

```
>>> i=2
>>> u=2.6
>>> type(u)
<class 'float'>
>>> type(i)
<class 'int'>
```

Variables prédéfinies

fond_pagebis

```
>>> import keyword
>>> print ("Liste des mots-cle :", keyword.kwlist)
Liste des mots-cle : ['False', 'None', 'True', 'and', 'as', 'assert', 'break'
    , 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally'
    , 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', '
    nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with
    ', 'yield']
```

Attention à ne pas utiliser ces termes comme nom de variables !

La calculatrice élémentaire

Quelques mots clés

- `int(a)` transforme la variable `a` en un nombre entier.
- `float(a)` transforme la variable `a` en un nombre décimal.
- `del(a)` efface la variable `a`.
- `print(a)`affiche la variable `a`.
- `break` permet de sortir d'une boucle.
- `continue` permet de relancer une boucle.

Comparateurs logiques

- `==,>,<,>=,<=,!=,...`

Opérateurs logiques

- `not,and,or`

Définir une fonction

Le fichier test.py

```
# test.py

def f(x):
    # f nom de la fonction
    # x entree de la fonction
    y=x**2+1
    return y # y sortie de la fonction

z=f(3)
print(z)
```

La sortie dans la console

```
>>> runfile('test.py', wdir='/home/fhubert/Enseignement/Agreg_interne/
    initiation_python/Cours_beamer')
10
```

- Attention à l'indentation dans la fonction

Définir une fonction

Le fichier test.py

```
# test.py

def f(x):
    # f nom de la fonction
    # x entree de la fonction
    y=x**2+1
    return y # y sortie de la fonction

z=f(3)
print(z)
```

A vous de jouer :

- Définissez la fonction $g(x) = x^3 + 2x - 4$
- Evaluez g en $x = 0$ et $x = -2$.

Définir une fonction

Autres exemples

```
# testb.py - plusieurs variables et sorties
def f(x,y): # entree de la fonction : x
    return x**2-y,y**2+1 #sortie de la fonction 2 termes

u,v=f(3,1)
print(u,v)
```

```
# testt.py : fonction caracteristique de ]4,+infty[
def f(x): # entree de la fonction : x
    if x>4:
        return 1
    return 0

print(f(2),f(10))
```

- Attention à l'indentation dans la fonction
- Dès que la fonction rencontre un `return`, elle s'arrête !

Test IF

```
# boucle IF
x=3
z=x**2-2*x+1
if z<1:
    y=0
elif z<3:
    y=1
else:
    y=2
print(y)
```

fond*pagebis*

- Attention à l'indentation dans la boucle
- Attention à ne pas oublier les `:`
- Attention, autant de `elif` que l'on veut mais un seul `else`.

A vous de jouer :

- Définir grâce à une boucle IF, la fonction qui à $x \in \mathbb{R}$ associe $x+1$ si $x \geq 1$ et 0 sinon.

Boucle FOR

```
# boucle FOR
u=[0.0]*10
u[0]=1
for n in range(9):
    u[n+1]=u[n]*2+3
print(u)
```

fond page agebis

- Attention à l'indentation dans la boucle
- Attention à ne pas oublier les :

A vous de jouer :

- Créer une fonction qui calcule les n premiers termes de la suite de Fibbonacci.

Boucle WHILE

```
# boucle WHILE
n=0
y=1
while y<20 and n<100:
    n=n+1
    y=y**2-3*y
print('n=',n,', y=',y)
```

fond_pagebis

- Attention à l'indentation dans la boucle
- Attention à ne pas oublier les :

A vous de jouer :

- Approchez à 10^{-6} près, le zéro de la fonction $f(x) = x^3 + 2x - 1$ sur l'intervalle $[0, 1]$ en utilisant l'agorithme de Dichotomie.

Objectifs du stage

fond$_p$agebis

La librairie math

https://docs.python.org/fr/3/library/math.html

```
from math import *
```

Le module propose les fonctions mathématiques et les constantes usuelles.

- Fonctions arithmétiques et de représentation
  `ceil, floor, gcd, ...`
- Fonctions logarithme et exponentielle
  `exp, log,pow,...`
- Fonctions trigonométriques
  `cos, sin, tan, acos, asin,...`
- Fonctions hyperboliques
  `cosh, sinh, tanh, acosh,...`
- Constantes
  `pi, e, tau, inf,nan`
- Fonctions spéciales
  `erf,...`

fond$_p$*agebis*

A vous de jouer :

- Effectuez le calcul suivant : $\cos(3\pi/6)$

La librairie numpy

https://docs.scipy.org/doc/numpy/

La librairie incourtournable du calcul scientifique !

```
import numpy as np
```

- Des variables prédéfinies
  `np.pi`
- Une gestion facile des tableaux/matrices
  `np.array, np.arange, np.linspace,np.zeros, np.ones, np.eye,np.dot...`
- Une librairie d'algèbre linéaire
  `np.linalg.det,np.linalg.solve,...`
- Inclus la plupart des fonctions présentes dans le module math

A vous de jouer :

- Effectuez le calcul suivant : $\cos(3\pi/6)$ en n'utilisant que la librairie numpy.

La librairie numpy

Définition des tableaux

```
# definition a la main
A=np.array([1,3,5,7])
B=np.array([[1,2],[3,4],[5,6]])

# Matrices predefinies
C=np.zeros((3,5))
D=np.eye(3,k=-1)
E=np.ones((3,2))
F=np.random.rand(4,4)
G=np.empty((2,2),'float')
H=np.identity(3)


# Definition via une boucle
M=np.zeros((3,3))
for i in range(3):
    for j in range(3):
        M[i,j]=1/(i+j+2)
```

L'accés aux éléments des tableaux se fait par :

`A[i,j], A[:,j],A[i,:],A[-1,1:3],...`

La librairie numpy

Définition d'une discrétisation

Deux méthodes : `np.arange, np.linspace`

```
# np.arange(xmin,xmax,pas)
H=np.arange(0,1,0.1)
```

Sortie :

```
>>> H
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9])
```

⚠ `xmax` n'est pas un point de la discrétisation

fond_pagebis

```
# np.linspace(xmin,xmax,N), N nombre de points
I=np.linspace(0,1,10)
```

Sortie :

```
>>> I
array([ 0.        ,  0.11111111,  0.22222222,  0.33333333,  0.44444444,
        0.55555556,  0.66666667,  0.77777778,  0.88888889,  1.        ])
```

⚠ Si N est le nombre de point de la discrétisation, le pas est égal à $\frac{1}{N+1}$.

La librairie numpy

A vous de jouer :

1. Créez le vecteur de discrétisation de l'intervalle $[0, 10]$ de pas 0.1.
2. Créez un vecteur de discrétisation de l'intervalle $[1, 3]$ à 20 éléments.
3. Créer la matrice $M \in \mathcal{M}_{3,3}$ telle que $M_{ij} = \frac{1}{i+j}$.

La librairie numpy

Opérations matricielles sur les tableaux

```
A=np.ones((2,2));B=np.array([[1,2],[1,2]])
# produit de deux matrices de meme taille
np.dot(A,B)
# array([[ 2.,  4.],[ 2.,  4.]])
A.dot(B)
# idem
np.linalg.det(A) # 0
# on modifie A pour le rendre inversible
A[0,0]=2
# inverse
np.linalg.inv(A) # array([[ 1., -1.],[-1.,  2.]])
x=np.array([1,0])
y=np.array([1,1])
# produit scalaire
np.inner(x,y) # 1
# Resolution du systeme lineaire
np.linalg.solve(A,x) # array([ 1., -1.])
# valeurs propres
np.linalg.eigvals(A) # array([ 2.61803399,  0.38196601])
#valeurs propres et vecteurs propres
np.linalg.eig(A)
# Transposee d'une matrice
A.T
```

La librairie numpy

A vous de jouer :

1. Créer une matrice A de taille 4×4 dont les coefficients sont choisis de façon aléatoire. Donner la partie symétrique de cette matrice $A_s = \frac{1}{2}(A + A^t)$.
2. Calculer le produit de A avec A_s.
3. Calculer le déterminant de A_s et son inverse si cela est possible.
4. Créer le vecteur $X = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$.
5. Créer la matrice tridiagonale B de taille 10×10 qui a des 2 sur la diagonale principale et des -1 sur les premières sur et sous diagonales.
6. Déterminer la solution du système linéaire $BX = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$.

La librairie numpy

A vous de jouer (encore !) :

Soit $A = \begin{pmatrix} \frac{1}{2} & \frac{1}{3} & \frac{1}{12} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{2} & \frac{1}{3} & \frac{1}{12} \\ \frac{1}{12} & \frac{1}{12} & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{12} & \frac{1}{12} & \frac{1}{2} \end{pmatrix}$.

1. A l'aide de la commade `numpy.linalg.eig` déterminez les valeurs propres de cette matrice. On vérifiera que 1 est une valeur propre associée au vecteur propre $\begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}^t$ et que les autres valeurs propres sont de module strictement inférieur à 1.
2. Créez une fonction qui à un entier n et un vecteur $X_0 \in \mathbb{R}^4$ associe $X_n = \frac{A^n X_0}{\|A^n X_0\|_1} = \frac{AX_{n-1}}{\|AX_{n-1}\|_1}$. Testez pour différentes valeurs de X_0 et des valeurs de n grandes votre fonction. Que remarquez-vous ?

fondpagebis

La librairie numpy

Tailles des tableaux

Les commandes `size,shape,reshape,...` en lien avec la taille des tableaux

```
B=np.array([[1,2],[3,4],[5,6]])
# Nombre d'\'el\'ements
np.size(B) # 6
# format
np.shape(B) # (3,2)
#redimensionnement
np.reshape(B,(2,3)) #array([[1, 2, 3],[4, 5, 6]])
```

fond_pagebis

Copies des tableaux

Si `A` est un `np.array`, la commande `B=A` crée juste un lien entre `B` et `A`.
Pour faire une copie, utiliser la commande `B=np.copy(A)`.

```
A=np.array([[1,2],[3,4]])
B=A
C=np.copy(A)
A[0,0]=5
print(B[0,0],C[0,0]) # 5, 1
```

La librairie numpy

Opérations termes à termes sur les tableaux

```
B=np.array([[1,2],[3,4],[5,6]])
C=2*np.ones((3,2))
B**2
# array([[ 1,  4],[ 9, 16], [25, 36]])
2*B+1
# array([[ 3,  5], [ 7,  9], [11, 13]])
B*C
# array([[  2.,   4.], [  6.,   8.], [ 10.,  12.]])
B/C
# array([[ 0.5,  1. ], [ 1.5,  2. ], [ 2.5,  3. ]])
np.cos(np.pi*B)
# array([[-1.,  1.], [-1.,  1.], [-1.,  1.]])
```

- Attention `B*C` n'effectue pas un produit matriciel
- Attention les fonctions `math.cos,...` ne prennent pas comme argument des `np.array`, il faut utiliser leurs analogues définies dans `numpy` :
  `np.cos, np.sin, np.exp, np.log, ....`

La librairie numpy

A vous de jouer :

1. Créez la discrétisation t de l'intervalle $[0, 1]$ de pas 0.1.
2. Créez la fonction $f : t \mapsto \frac{1}{1-t}$. $\quad \text{fond}_p agebis$
3. Créez le vecteur des images par f du vecteur t.

La librairie numpy

Opérations globales sur les tableaux

```
B.max() # max global 6
B.max(0) # max sur les colonnes array([5, 6])
B.max(1) # max sur les lignes array([2, 4, 6])
```

Même chose pour `B.min`, `B.mean`, `B.sum`, `B.prod`.

Diagonales

```
np.diag(A) # diagonale de A
np.diag(np.diag(A)) # matrice diagionale de diagonale identique a A
```

Normes

```
np.linalg.norm(A) # norme induite associee a la norme 2
np.linalg.norm(A,np.inf) # norme induite associe a la norme infinie
np.linalg.norm(A,1) #  norme induite associee a la norme 1
```

Autres possibilités,voir
https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.norm.html

La librairie matplotlib.pyplot

https://matplotlib.org/users/pyplot_tutorial.html

```
plt.figure(1)
plt.clf()
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.savefig('plot1.png')
```

fond_pagebis

```
plt.figure(2)
plt.clf()
x=np.arange(0,1.01,0.1)
y=x**2
plt.plot(x,x,'r')
plt.plot(x,y,'b',LW=2)
plt.legend(['x','x^2'],loc=2)
```

La librairie matplotlib.pyplot

https://matplotlib.org/users/pyplot_tutorial.html

```
plt.loglog(x,x,'r')
plt.loglog(x,y,'b',lw=2)
plt.legend(['x','x^2'],loc=2)
plt.title('Echelle logarithmique')
```

Autres commandes :

plt.semilogx,plt.semilogy

fond_pagebis

- Attention, sous certaines configurations des IDE, pour faire apparaitre les graphes il faut rajouter la commande plt.show()
- Remarque la librairie matplotlib.pyplot est similaire à la librairie matplotlib.pyplot qui est conseillée.

La librairie matplotlib.pyplot

1. Créez le vecteur de discrétisation de $I = [0, 5]$ de pas 0.1.
2. Soit $y_0 = 0.1$. Créez la fonction

$$y : t \mapsto \frac{1}{\frac{1}{b} + \left(\frac{1}{y_0} - \frac{1}{b}\right) e^{-at}}$$

fond_page.bis

3. Tracez sur une même figure, le graphe de y sur l'intervalle I pour différents couples de (a, b) : $(1, 10)$, $(2, 10)$, $(1, 20)$, $(2, 20)$. On changera de style pour chacune des courbes, on pensera à rajouter une légende.

La librairie matplotlib.pyplot

Un exemple plus complet

```
t = np.linspace(0, 2 * np.pi, 400)
x=np.cos(t)
y=np.sin(t)
z= np.sin(t ** 2)

plt.close('all')
fig=plt.figure(1);plt.clf()

plt.subplot(3,1,1)
plt.plot(t,z,'r-.',lw=3)
plt.xlabel('t')
plt.ylabel('sin(t^2)')

plt.subplot(3,1,2)
plt.plot(t,x,'m')
plt.plot(t,y,'c')
plt.legend(['cos','sin'])

plt.subplot(3,1,3)
plt.plot(x,y,'+k')
plt.axis([-1,1,-1,1])

plt.suptitle('Gros titre')
```

Objectifs du stage

fond_pagebis

Pour aller plus loin

Les entrées-sorties

```
>>> n=input("Entrer un entier : ")
Entrer un entier : 10
>>> n # Attention, n est une chaine de caractere
'10'
>>> n0=int(n);n0**2 # int transforme cette chaine en entier
100
 >>> x=input("Entrer un decimal : ")
Entrer un decimal : 10.0
>>> x # Attention, x est une chaine de caractere
'10.0'
>>> x=float(x);x**2 # float transforme cette chaine en entier
100.0
>>> print(n0,x)
10,100.0
```

Les modules personnels

Créer sa librairie de fonctions

```
# malibrairie.py
def f(y):
    return y**2
def g(x):
    return 2*x-1
```

Deux façons de faire appel à cette librairie

```
import malibrairie

print(malibrairie.f(2),malibrairie.g(3))
```

ou

```
from malibrairie import f,g

print(f(2),g(3))
```

Définir une fonction

Autres exemples

```
# tests.py , arguments non precises
def produit(*arg):
    """ Retourne le produit des arguments """ # commentaire
    P=1
    for n in arg:
        P*=n
    return P

z=produit(1,2,3,4)
print(z)
help(produit)   # ecrit le commentaire
```

Définir une fonction

Autres exemples

```
#  variable x uniquement d\'efinie dans la fonction
def f(y):
    return x+y
x=2
print(f(3),x)
```

fond_page_agebis

```
#  variable x uniquement definie dans la fonction
def g(y):
    x=3
    return x+y
x=2
print(g(3),x)
```

Un exemple classique : les suites récurrentes

Commenter ce programme !

$$u_{n+1} = f(u_n)$$

pour

$$f(x) = x^2$$

fig_escargot1_b.png

```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**2

def visualise(x0,fonction,p,q,xmin,xmax,ymin,ymax):
    l=np.size(x0)
    xx=np.linspace(xmin,xmax,100)

    plt.figure(q)
    plt.clf()
    plt.plot(xx,xx,'-.c',lw=3)
    plt.plot(xx,fonction(xx),'-.m',lw=3)

    for s in range(l):
        x=np.zeros(p+1);yy=np.zeros((2*p,2))
        x[0]=x0[s]
        for i in range(p):
            x[i+1]=fonction(x[i])
            yy[2*i,0]=x[i];yy[2*i,1]=x[i+1];yy[2*i
                +1,0]=x[i+1];yy[2*i+1,1]=x[i+1]
        plt.plot(yy[:,0],yy[:,1])
        plt.axis([xmin,xmax,ymin,ymax])

visualise(np.array([0.5,0.8,1.1]),f,5,1, 0,4,0,4)
```

Objectifs du stage

1. Un exemple de code python
2. Python : une maxi calculatrice
   - Les variables
   - Les fonctions
   - Les boucles et les tests
3. Les modules Python
   - Les librairies classiques
     - Fonctions mathématiques : math
     - Calcul scientifique : numpy, scipy,SymPy
     - Graphiques : matplotlib.pyplot
4. Pour aller plus loin
   - Les entrées-sorties
   - Les modules personnels
   - Les fonctions
5. **Exercices pour aller plus loin**

fond$_p$*agebis*

Exercices pour aller plus loin

- Série de Fourier
- Graphes

fond$_p$*agebis*