

Introduction à la programmation

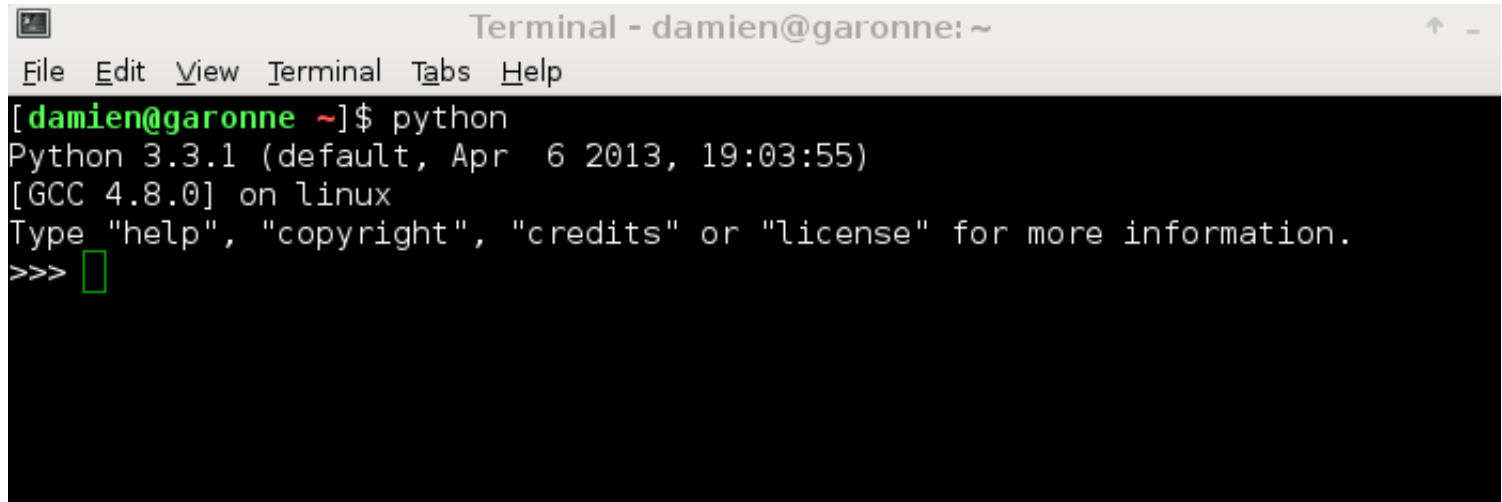
Python

2013



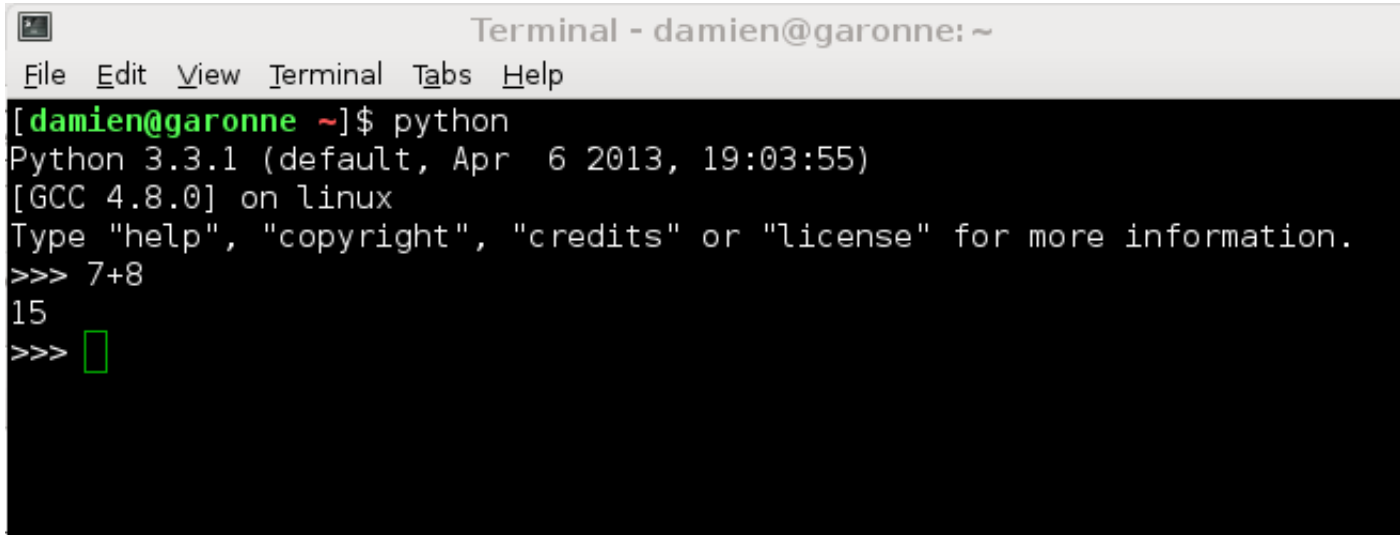
Damien Rohmer

Premier "programme"

A terminal window titled "Terminal - damien@garonne: ~" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows the execution of the "python" command, displaying the Python version (3.3.1), GCC version (4.8.0), and the operating system (linux). It also provides instructions on how to get help, copyright, credits, or license information. The prompt ">>>" is followed by a green cursor.

```
Terminal - damien@garonne: ~
File Edit View Terminal Tabs Help
[damien@garonne ~]$ python
Python 3.3.1 (default, Apr  6 2013, 19:03:55)
[GCC 4.8.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Premier "programme"



```
Terminal - damien@garonne: ~  
File Edit View Terminal Tabs Help  
[damien@garonne ~]$ python  
Python 3.3.1 (default, Apr 6 2013, 19:03:55)  
[GCC 4.8.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> 7+8  
15  
>>> 
```

Commandes

Notion de variables:

```
a=7
```

```
b=2
```

```
a+8+b*b
```

```
> 17
```

a est une **variable** (qui vaut 7)

b est une **variable** (qui vaut 2)

Commandes

Notion de variables:

```
a=4  
b=a+1  
b=b+2  
  
print(b)
```

> 7

Commandes

Affichage à l'écran:

```
print("le resultat de 2+2 vaut",2+2)
```

> le resultat de 2+2 vaut 4

Commandes

Types de variables:

`a=5` ← a est un nombre (entier)
`b=4.12` ← b est un nombre (à virgule)
`c="du texte"` ← c est un texte

`a+b` OK

~~`a+c`~~

unsupported operand type(s) for +: 'int' and 'str'

Commandes

Types de variables:

```
a=7  
b=2  
c=-b/a;  
print("la solution de l'equation ",a,"x + ",b,"=0 vaut",c)
```

Commandes

Variable nombre/texte:

```
mon_texte_1="4+7"
mon_texte_2="2+2"
valeur_1=4+7
valeur_2=2+2

mon_texte_3=mon_texte_1+mon_texte_2
valeur_3=valeur_1+valeur_2

print(mon_texte_3)
print(valeur_3)
```

ceci est du texte

ceci est un nombre

```
> 4+72+2
> 15
```

Commandes

Variable nombre/texte:

transforme un nombre en texte
(str=string)

```
variable_nombre=4+8  
variable_texte=str(variable_nombre);  
  
variable_texte=variable_texte+"78"  
  
print(variable_texte)
```



> 1278

L'aide

Pour obtenir de l'aide sur une fonction:

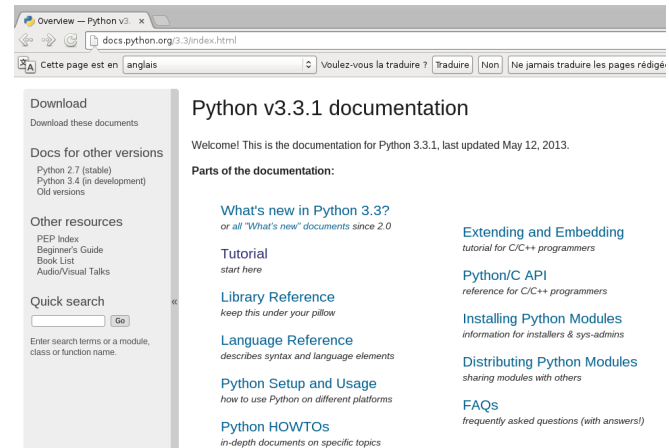
help(*nom_fonction*)

ex. help(pow)

Site web:

<http://docs.python.org/2/index.html>

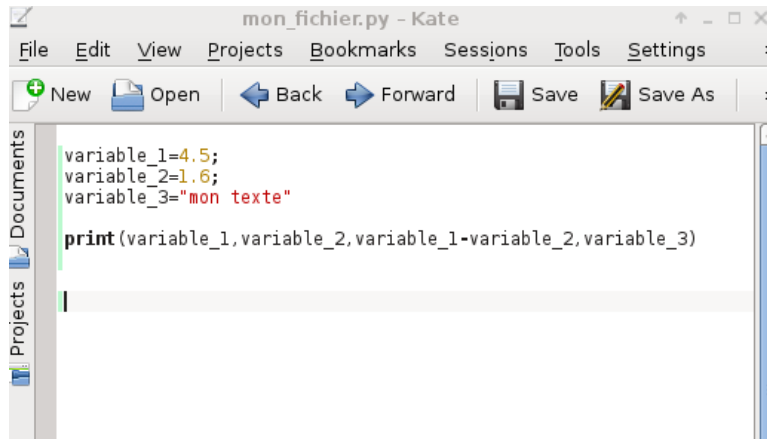
<http://docs.python.org/3/index.html>



Ecriture dans un fichier

Ecrire ligne à ligne est fastidieux ...

On écrit d'abord dans un fichier texte

A screenshot of a text editor window titled 'mon_fichier.py - Kate'. The window shows a menu bar (File, Edit, View, Projects, Bookmarks, Sessions, Tools, Settings) and a toolbar with icons for New, Open, Back, Forward, Save, and Save As. The code in the editor is:

```
variable_1=4.5;
variable_2=1.6;
variable_3="mon texte"

print(variable_1,variable_2,variable_1-variable_2,variable_3)
```

(.py = fichier texte lisible par Python)

On lance Python sur le fichier

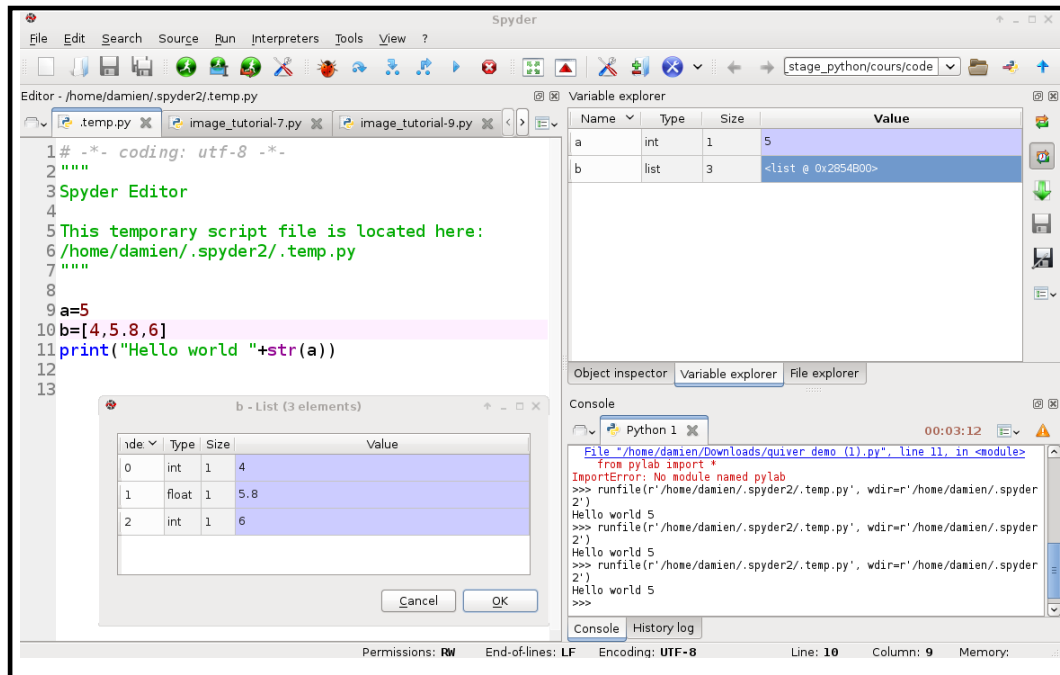
```
[damien@damien_pc ~/work/2012_2013_teaching
ython/cours/code]$ python mon_fichier.py
4.5 1.6 2.9 mon texte
```

Editeur Python

Editeur de texte (attention à l'indentation)

- Linux: Kate
- Window: par défaut, pyscripter

Editeur type Matlab: **Spyder**



Python: le langage

Création en 1990 (C ~ 1973)

Scripts, manipulation texte, pas de scientifique

Module Numpy en 2005

Developpement du calcul scientifique

Python 2.0 en 2000

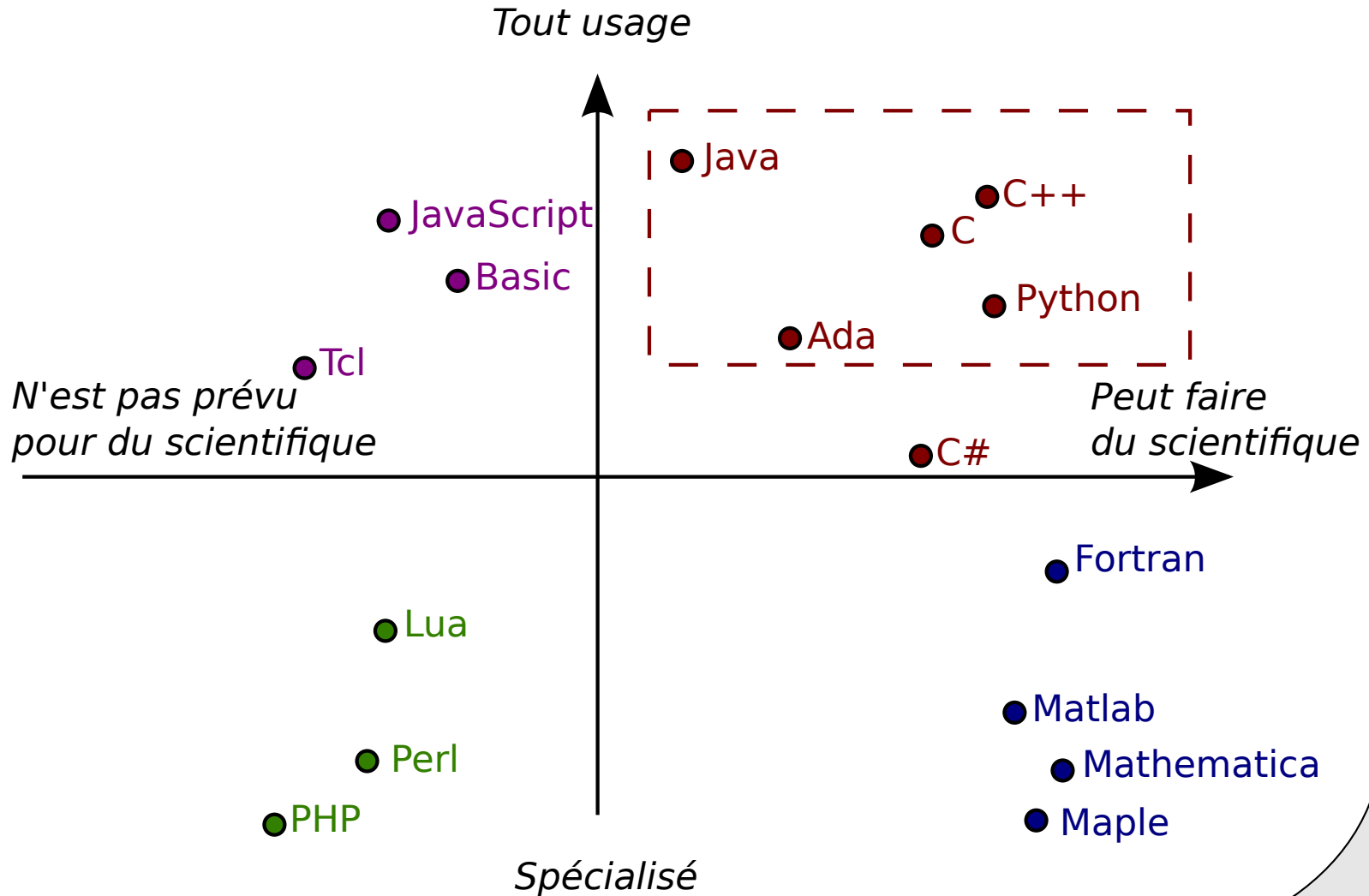
Python 3.0 en 2009



Python devient un acteur majeur du monde du calcul scientifique

- beaucoup de modules (scientifique, visualisation, etc)
- lisible
- simple à écrire
- langage haut niveau
- potentiellement optimisable

Python: positionnement



Python: positionnement

Simple, Lisible



● Python

● Java

● Ada

● C

● C++

Complexe

Plan

1: Bases du langage

2: Librairie mathématique et affichage graphique

3: Entrées/sorties, utilisation de fichiers

Plan

1: Bases du langage

Conditions: if/else

Listes

Creation de listes, iterations "for"

Fonctions

Manipulation de fonctions réelles

Calcul intégral

Boucles for

Boucles while

Recherche de zéros par dichotomie

2: Librairie mathématique et affichage graphique

3: Entrées/sorties, utilisation de fichiers

Plan

1: Bases du langage

2: Librairie mathématique et affichage graphique

Numpy: array

Calculs géométriques

Matplotlib: affichage de courbes

Dérivée discrète et équations différentielles

Pendule oscillant

Matrices

Résolution système linéaires

Diagonalisation

Racines de polynomes

Affichage de matrices et d'images

Affichage de fonctions 2D

Fractales

Champ de vecteurs

Mécanique des fluides

3: Entrées/sorties, utilisation de fichiers

Plan

1: Bases du langage

2: Librairie mathématique et affichage graphique

3: Entrées/sorties, utilisation de fichiers

Chaine de caractères

Lecture et écriture de fichiers

Visualisation de données scientifiques

Formattage de fichiers de notes

Communication avec programmes externes

Conditions: si, sinon

Condition *if*

"si"

```
a=5
```

```
b=5
```

```
if a*b > 22:                (si a fois b est plus grand que 22)  
    print("j'affiche ce message")
```

```
a,b=8,9
```

```
c=0
```

```
if a+b-b*b<=3:  
    c=c+3*a
```

```
if c/10>c*c:  
    print("j'affiche ce message")
```

Condition *if*

"si"

```
a=5
```

```
b=5
```

: début d'un bloc de traitement

```
if a*b > 22:
```

```
    print("j'affiche ce message")
```

espace => bloc d'instructions

Condition *if*

"si"

```
x=12.2
```

```
if x>=0 and x<5:  
    print("intervalle [0,5[")
```

```
if x>2 or x<0:  
    print("intervalle [-inf,0[ U ]2;+inf[")
```

Condition *if* / *else*

"si / sinon"

```
a=5
b=4

if a*b > 22:
    print("j'affiche ce message")
else:
    print("ceci est un autre message")
```

si a fois b est supérieur à 22
alors "j'affiche ce message"

sinon
j'affiche "ceci est un autre message"

Condition *if / else*

"si / sinon"

Cas particulier du test d'égalité

```
a=5
if(a==6):
    print("a vaut 6")
else
    print("a ne vaut pas 6")
```

symbole ==
test d'égalité

si *a est égale à 6*
alors affiche "a vaut 6"
sinon
affiche "a ne vaut pas 6"

Math	Code
$a := b$	<code>a=b</code> affectation
$a = b$	<code>a==b</code> test d'égalité (vaut <i>vrai</i> ou <i>faux</i>)

Condition *if / else*

"si / sinon"

Conditions if/elif/else

elif = else, if (/ sinon, si ...)

```
if x>=0 and x<2:  
    print("intervalle [0,2[")  
elif x<4:  
    print("intervalle ]-inf,0[ U [2,4[")  
elif x<=5:  
    print("intervalle ]-inf,0[ U [4,5]")  
else:  
    print("intervalle ]5,+inf[")
```

Condition *if* imbriqués

Quelle courbe dessine y si x varie ?

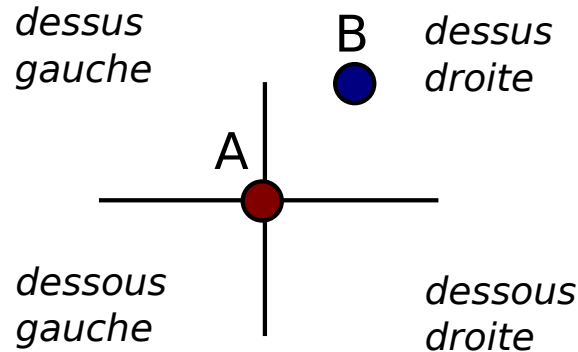
```
if x>=0:
    if x<1:
        y=x*x
    else:
        y=x
else:
    if x<-1:
        y=-x*x
    else:
        y=-x-2
```

Condition *if* imbriqués

Application:

Soit 2 points $A=(x_1,y_1)$ et $B=(x_2,y_2)$

Indiquer si A est au dessus/dessous de B
gauche/droite



Les listes d'éléments

Ensemble éléments: Listes

crochets [...] indiquent une liste

```
vecteur=[1,7,5,9,-4,3]
```

```
print(vecteur[0])
```

```
print(vecteur[1])
```

```
print(vecteur[2])
```

```
print(vecteur[0]+vecteur[4])
```

```
vecteur[2]=vecteur[4]*vecteur[5]
```

```
print(vecteur)
```

contenu:

1	7	5	9	-4	3
---	---	---	---	----	---

indices:

[0]	[1]	[2]	[3]	[4]	[5]
-----	-----	-----	-----	-----	-----

Ensemble éléments: Listes

```
vecteur=[1,7,5,9,-4,3]  
print(vecteur[6])
```

IndexError: list index out of range

contenu:	1	7	5	9	-4	3	??
indices:	[0]	[1]	[2]	[3]	[4]	[5]	[6]

Ensemble éléments: Listes

Une liste peut contenir des mots

```
vecteur=[ "pomme", "poire", "banane", "peche"]  
  
vecteur[0]=vecteur[1]+" "+vecteur[2]  
print(vecteur)
```

"pomme"

[0]

"poire"

[1]

"banane"

[2]

"peche"

[3]

Ensemble éléments: Listes

Une liste peut contenir différents types

```
vecteur_mixte=[1.45,-7,"un torchon",1,"une serviette"]  
  
print(vecteur_mixte[0])  
print(vecteur_mixte[2])  
print(vecteur_mixte)
```

1.45	-7	"un torchon"	1	"une serviette"
[0]	[1]	[2]	[3]	[4]

Ensemble éléments: Listes

Ajouter des éléments dans une liste

```
vec=[4,5,6]  
print(vec)  
vec.append(7);  
vec.append(8);  
print(vec)
```

Ensemble éléments: Listes

Supprimer des éléments dans une liste

```
vec=[4, -1, 5, 7, 12]  
print(vec)  
del(vec[0])  
print(vec)  
del(vec[2])  
print(vec)
```

Ensemble éléments: Listes

Créer une "liste" particulière

```
a=range(4,9)
b=range(8,-2,-3)
print(a[0],a[1],a[2],a[3],a[4])
print(b[0],b[1],b[2],b[3])
```

a

4	5	6	7	8
---	---	---	---	---

b

8	5	2	-1
---	---	---	----

range(debut,fin,[*increment*])



stop 1 élément
avant fin

Ensemble éléments: Listes

Nombre d'éléments d'une liste

```
vec1=[1,4,8,9]
vec2=["canard",7.45,"poireaux"]

longueur_1=len(vec1)
print(longueur_1)

print(len(vec2))
```

Ensemble éléments: Listes

Indexation inverse

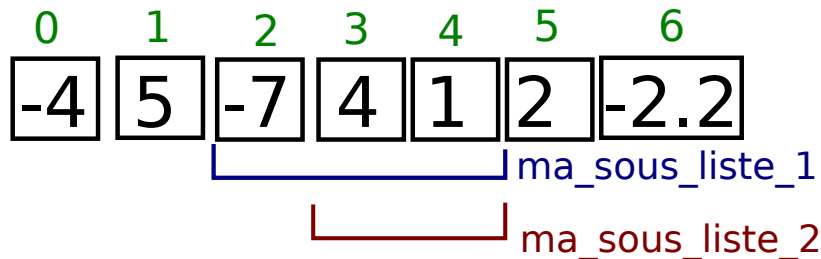
```
vec=[1, -4, 7, 2, 6, 8, 3]
```

```
print(vec[-1])
```

```
print(vec[-2])
```

Sous partie d'une liste

```
ma_liste=[-4,5,-7,4,1,2,-2.2]  
ma_sous_liste_1=ma_liste[2:5]  
ma_sous_liste_2=ma_liste[3:-2]  
  
print(ma_sous_liste_1)  
print(ma_sous_liste_2)
```



Trier une liste

```
ma_liste=[4,1,-7,9,5,12,-3]
ma_liste_triee=sorted(ma_liste)

print(ma_liste_triee)
```

```
ma_liste=["velo","cheval","nenuphar","pilote"]
ma_liste_triee=sorted(ma_liste)

print(ma_liste_triee)
```

```
ma_liste=[4,"cheval",7.8]
sorted(ma_liste)
```

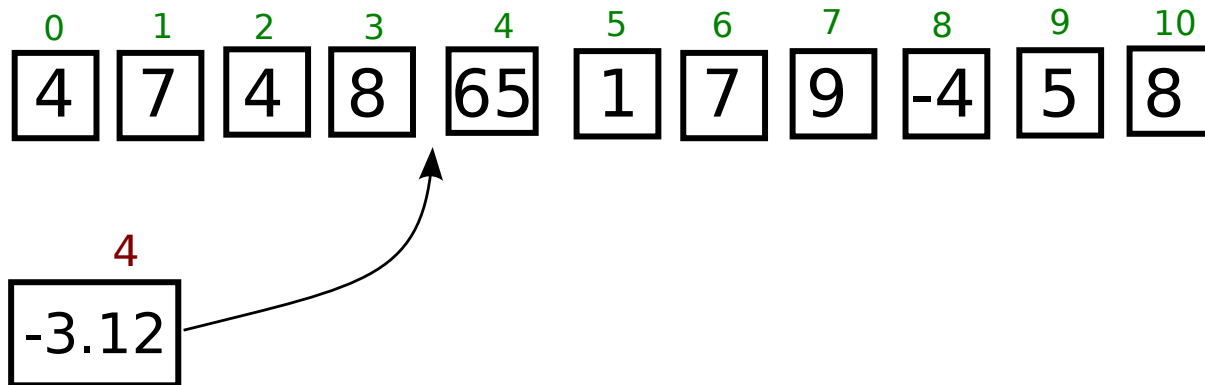
*unorderable types
str() < int()*

Compter nombre d'occurences

```
ma_liste=[7,8,4,-1,4,8,-2,-1,1]  
nombre_de_huit=ma_liste.count(8)  
  
print(nombre_de_huit)
```

Insérer un élément dans une liste

```
ma_liste=[4,7,4,8,65,1,7,9,-4,5,8]  
ma_liste.insert(4,-3.12)
```



Supprimer une valeur

```
ma_liste=[4,7,4,8,65,1,7,9,-4,5,8]  
ma_liste.remove(8)
```

Ne supprime qu'une valeur (la première trouvée)



Différent de: **del**(*ma_liste*[*k*])

supprime le kème élément (indice)

Liste de listes

```
triangle=[[0,0,0],[1,-0.1,1.1],[0,1,0.5]]  
  
print(triangle[0])  
print(triangle[1])  
print(triangle[2])  
print(triangle[1][2])
```

Application sur les listes

Soit `ma_liste=[1,2,3,4,5,6,7,8,9]`

Supprimer le deuxième élément de la liste

Afficher le troisième élément de la nouvelle liste

Insérer un 2 en quatrième position de la nouvelle liste

Soustraire 3 à l'avant dernier élément

Afficher la nouvelle liste

Trier la nouvelle liste puis l'afficher

Application sur les listes

Soit `ma_liste=[1,2,3,4,5,6,7,8,9]`

Supprimer le deuxième élément de la liste

Afficher le troisième élément de la nouvelle liste

Insérer un 2 en quatrième position de la nouvelle liste

Soustraire 3 à l'avant dernier élément

Afficher la nouvelle liste

Trier la nouvelle liste puis l'afficher

`[1, 2, 3, 4, 5, 5, 6, 7, 9]`

```
ma_liste=[1,2,3,4,5,6,7,8,9]
del(ma_liste[1])
print(ma_liste[2])
ma_liste.insert(3,2)
ma_liste[-2]-=3
print(ma_liste)
ma_liste=sorted(ma_liste)
print(ma_liste)
```

Ensemble éléments: Listes

Exercice:

Combien y a t'il de nombres entre -525 et 640 (inclus)
en comptant de 5 en 5 ?

Egalité de liste

L'affectation de liste référence la même entité

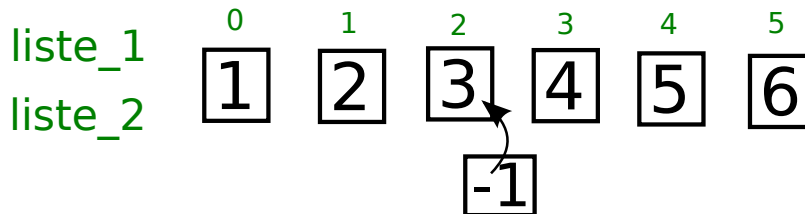
```
liste_1=[1,2,3,4,5,6]
liste_2=["a","b","c","d"]

liste_2=liste_1

print(liste_2)

liste_2[3]=-1

print(liste_2)
print(liste_1)
```



Copie de liste

Si l'on souhaite dupliquer une liste, on appelle explicitement `list(nom_liste)`

```
liste_1=[1,2,3,4,5,6]
liste_2=["a","b","c","d"]

liste_2=list(liste_1)

print(liste_2)

liste_2[3]=-1

print(liste_2)
print(liste_1)
```

création
d'une copie de la liste

	0	1	2	3	4	5
liste_1	1	2	3	4	5	6
	0	1	2	3	4	5
liste_2	1	2	3	-1	5	6

Itération sur les listes

le mot clé "*for*"

Créer des listes

$$L = \{f(k) | k \in \llbracket 0, N \rrbracket\}$$

Exemple pour $f(k) = (k - 2)^2$

En *français*:

`L = (k-2)^2 pour k variant dans [0,N[`

En *code*

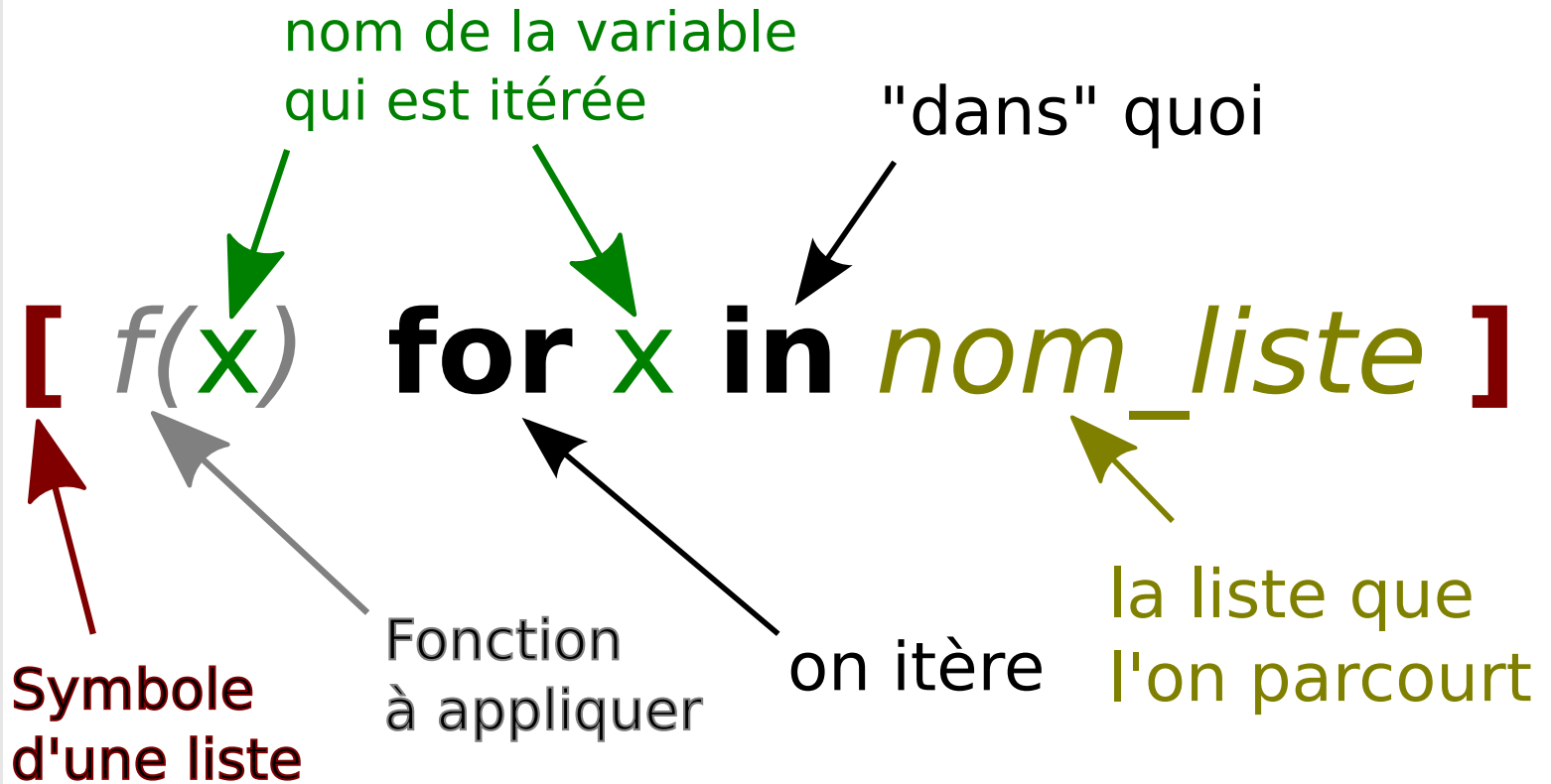
`L=[(k-2)**2 for k in range(0,N)]`

(** :puissance)

`N=4`

`L=[(k-2)**2 for k in range(0,N)]
print(L)`

La boucle "pour"



Application

Calculus:

$$L = \{a_k \mid k \in \llbracket -N, N \rrbracket\}$$

$$a_k = k\sqrt{|k-1|}/4$$

$(|x| : \text{abs}(x))$

Application

Calculer:

$$L = \{a_k \mid k \in \llbracket -N, N \rrbracket\}$$

$$a_k = k \sqrt{|k - 1|} / 4$$

$(|x| : \text{abs}(x))$

```
L=[(k*abs(k-1)**0.5)/4 for k in range(-N,N)]
```

Application

$$L = \{a_k \mid k \in \llbracket -N, N \rrbracket\}$$

$$a_k = k\sqrt{|k-1|}/4$$

```
import matplotlib.pyplot as plt  
  
L=[(k*abs(k-1)**0.5)/4 for k in range(-N,N)]  
  
plt.plot(L)  
plt.show()
```



affichage (plot=dessine, show=montre à l'écran)

Application

$$L = \{a_k \mid k \in \llbracket -N, N \rrbracket\}$$

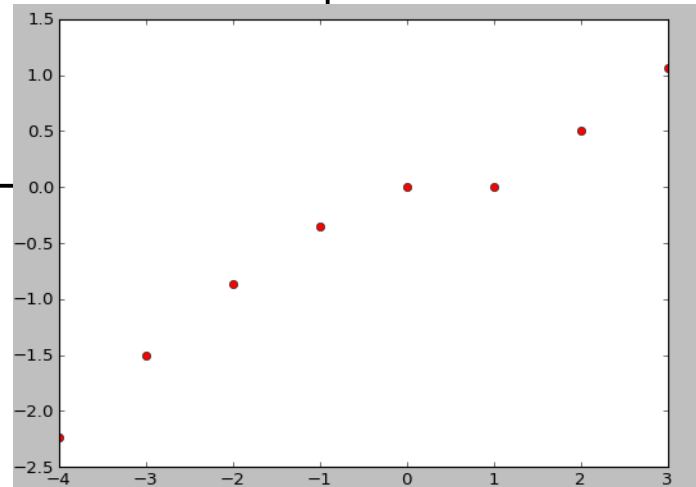
$$a_k = k\sqrt{|k-1|}/4$$

Avec les bonnes abscisses + echantillons

```
import matplotlib.pyplot as plt  
L=[(k*abs(k-1)**0.5)/4 for k in range(-N,N)]  
abscisses=range(-N,N)  
plt.plot(abscisses,L,"ro")  
plt.show()
```

en rouge

dessiner des ●



Les fonctions

Les fonctions

Remarque:

$$L = \{f(k) \mid k \in \llbracket k_0, k_N \rrbracket\}$$

$$f(k) = \dots$$

```
L=[(k*abs(k-1)**0.5)/4 for k in range(-N,N)]
```

compliqué ... peu lisible!

On souhaiterait écrire:

```
L=[f(k) for k in range(k0,kn)]
```

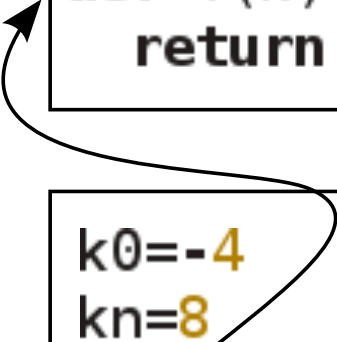
avec $f(k) = \dots$

Les fonctions

$$L = \{f(k) \mid k \in \llbracket k_0, k_N \llbracket \}$$

$$f(k) = k\sqrt{|k-1|}/4$$

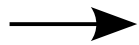
```
def f(k):  
    return k*abs(k-1)**0.5/4
```



```
k0=-4  
kn=8  
L=[f(k) for k in range(k0, kn)]
```

Les fonctions

def nom_fonction(argument):



Faire quelque chose ...

return valeur

```
def f(k):  
    return k*abs(k-1)**0.5/4  
  
k0=-4  
kn=8  
L=[f(k) for k in range(k0,kn)]
```

Les fonctions

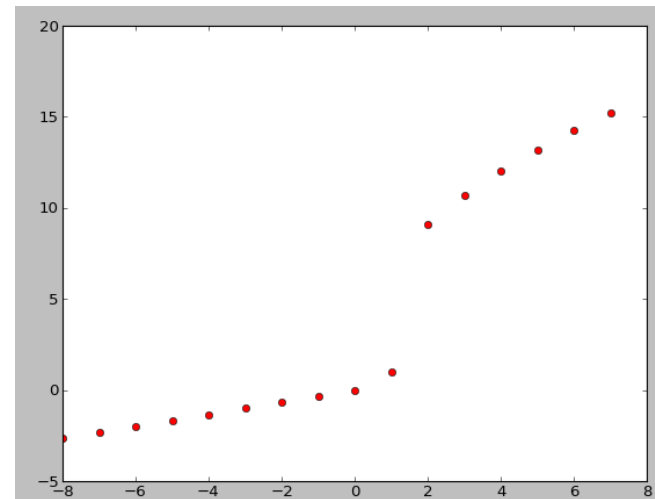
Les fonctions peuvent être *compliquées*

$$f(k) = 5\sqrt{k} + 2 \quad k \geq 2$$

$$f(k) = k^2 \quad k \in]-1, 2[$$

$$f(k) = k/3 \quad k \leq -1$$

```
def f(k):  
    if(k>=2):  
        return 5*k**0.5+2  
    if(k<=-1):  
        return k/3  
    if(-1<k<2): #(optionnel)  
        return k**2  
  
L=[f(k) for k in range(-8,8)]  
plt.plot(range(-8,8),L,"ro")  
plt.show()
```



Les fonctions

Les fonctions peuvent prendre des paramètres

$$f(k) = ak^2 + b$$

```
def f(k,a,b):  
    return a*k**2+b
```

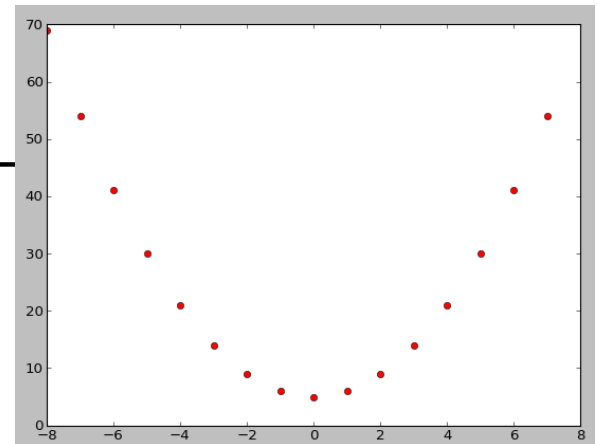
```
a=1
```

```
b=5
```

```
L=[f(k,a,b) for k in range(-8,8)]
```

```
plt.plot(range(-8,8),L,"ro")
```

```
plt.show()
```



Les fonctions

Les fonctions peuvent être vectorielles

 $\mathbb{R}^3$ $\mathbb{R}^2$

$$f : (x, y, z) \mapsto (x^2 + y, xy)$$

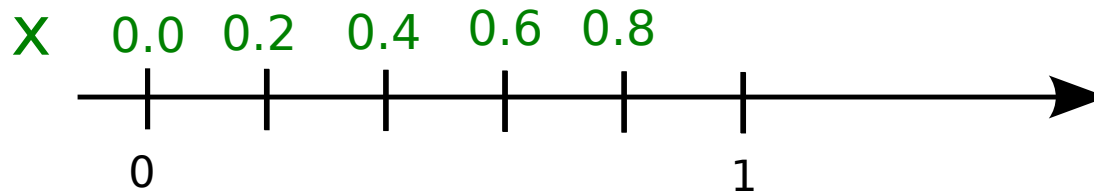
```
def f(x, y, z):  
    return [x**2+y, x*y]
```

```
v=f(1.5, 2.2, -1.1)  
print(v)
```

Manipulation de fonctions réelles

Echantillonner dans $\mathbb{R}$

Calculer et stocker N valeurs
également réparties sur $[0,1[$



```
N=10  
x=[k/N for k in range(0,N)]  
print(x)
```

Echantillonner dans $\mathbb{R}$

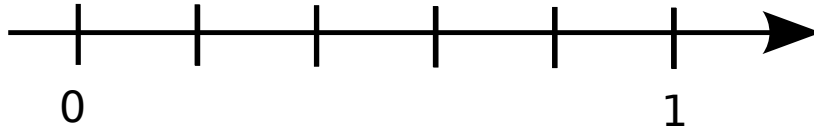
Calculer et stocker les N échantillons de $f(x)$

$$f(x) = x(x^2 - 1)$$

$$x \in [0, 1[$$

$y[0]$ $y[1]$ $y[2]$ $y[3]$ $y[4]$

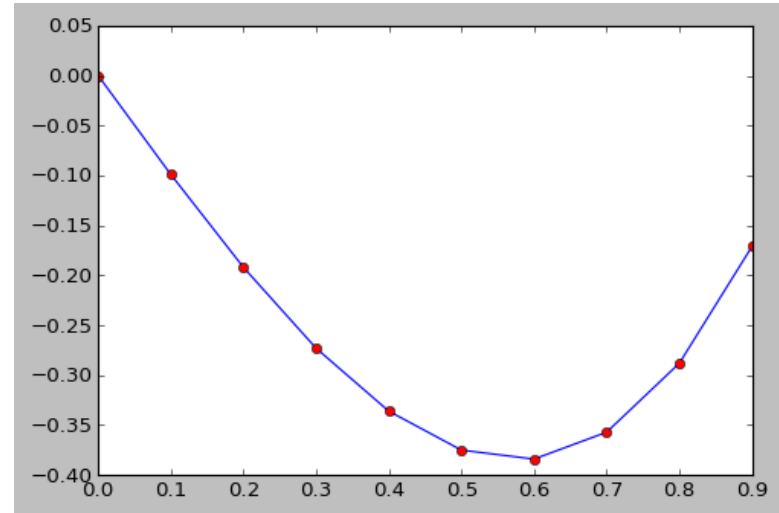
0.0 0.2 0.4 0.6 0.8



```
def f(x):  
    return x*(x**2-1)
```

```
N=10  
x=[k/N for k in range(0,N)]  
  
y=[f(x_k) for x_k in x]
```

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



Echantillonner dans $\mathbb{R}$

Calculer N échantillons de f pour $x \in [a, b[$

$$f(x) = \cos(2x)$$

$$[a, b[= [-2.3, 4.1[$$

```
from math import cos

def f(x):
    return cos(2*x)

N=30

#vecteur sur [0,1[
u=[k/N for k in range(0,N)]

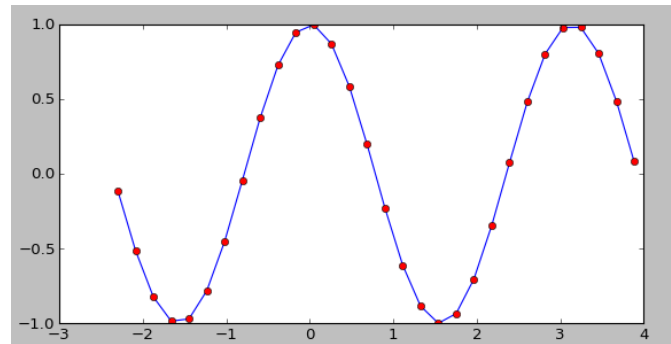
#vecteur sur [a,b[
a,b=-2.3,4.1
x=[u_k*(b-a)+a for u_k in u]

#f(x) pour x sur [a,b[
f=[f(x_k) for x_k in x]

plt.plot(x,f)
plt.plot(x,f,"ro")
plt.show()
```

$$\left[0, \frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N}\right]$$

$$(b-a) \left[0, \frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N}\right] + a$$



Application

La diode possède une caractéristique s'exprimant sous cette forme

$$I = I_0 \left(\exp \left(\frac{V}{V_T} \right) - 1 \right)$$
$$V_T = \frac{kT}{q}$$

$$I_0 = 100 \text{ nA}$$
$$q = 1.6 \cdot 10^{-19} \text{ C}$$
$$k = 1.38 \cdot 10^{-23} \text{ J/K}$$

Tracer la caractéristique pour $T=25^\circ$ et $T=65^\circ$
(298K) (338K)

$$V \in [-0.1, 0.4] \text{ V}$$

Rem: Pour exponentielle

```
from math import exp  
  
exp(x)
```

Application

$$I = I_0 \left(\exp \left(\frac{V}{V_T} \right) - 1 \right) \quad V_T = \frac{k T}{q}$$

$$I_0 = 100 \text{ nA}$$

$$q = 1.6 \cdot 10^{-19} \text{ C}$$

$$k = 1.38 \cdot 10^{-23} \text{ J/K}$$

$$V \in [-0.1, 0.4] \text{ V}$$

$$(298\text{K})$$

$$(338\text{K})$$

```
def f(V,T):
```

```
    #constantes
```

```
    k=1.38e-23
```

```
    q=1.6e-19
```

```
    I0=100e-9
```

```
    # "thermal voltage"
```

```
    Vt=k*T/q
```

```
    # courant
```

```
    I=I0*(exp(V/Vt)-1)
```

```
    return I
```

```
N=100
```

```
Vmin=-0.1
```

```
Vmax=0.4
```

vecteur sur [0,1[

```
u=[k/N for k in range(0,N)]
```

vecteur sur [V_{min}, V_{max}[

```
V=[(Vmax-Vmin)*u_k+Vmin for u_k in u]
```

```
# courant a 25 degrees
```

```
I_1=[f(v_k,298) for v_k in V]
```

```
# courant a 65 degrees
```

```
I_2=[f(v_k,338) for v_k in V]
```

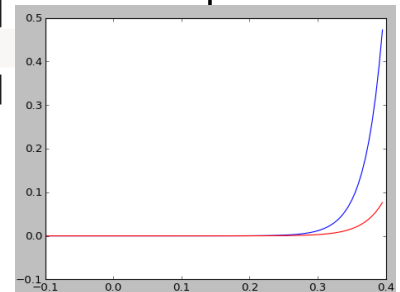
```
plt.plot(V,I_1,"b")
```

```
plt.plot(V,I_2,"r")
```

```
plt.show()
```

```
print(I)
```

```
print(V)
```



Fonctions disponibles

Somme des éléments

$$S = \sum_k x_k$$

```
vec=[1,7,8,4,5]  
S=sum(vec)
```

Application:

$$n = \left(\sum_k x_k^2 \right)^{1/2}$$

```
vec=[1,7,8,4,5]  
n=sum([x_k**2 for x_k in vec])**0.5
```

Min/Max des éléments

```
vec=[1,7,8,4,-2,5]  
a=max(vec)  
b=min(vec)  
print(a)  
print(b)
```

Il existe un élément ...

Soit $(a_k)_{k \in [0, N[}$

Question: $\exists k \in [0, N[, a_k = 4$

En français:

Il existe au moins un élément tel que a_k égale 4
pour k variant entre $[0, N[$

En code:

```
any(x==4 for x in vec)
```

True

(type booléen)

False

Il existe un élément ...

Exemple:

```
vec=[1,7,8,4,-2,5]  
est_ce_vrai = any(x==4 for x in vec)  
  
print(est_ce_vrai)
```

```
vec=[1,7,8,4,-2,5]  
est_ce_vrai = any(x>9 for x in vec)  
  
print(est_ce_vrai)
```

Tous les éléments ...

Soit $(a_k)_{k \in [0, N[}$

Question: $\forall k \in [0, N[, a_k < 12$

En français:

Tous les a_k sont inférieurs à 12
pour k variant entre $[0, N[$

En code:

```
all(x < 12 for x in vec)
```

True

(type booléen)

False

Application

```
v1=[1,0,0]
v2=[0,1,0]
v3=[1/(3**0.5),1/(3**0.5),1/(3**0.5)]
v4=[1/(2**0.5),0/(2**0.5),1/(2**0.5)]

ensemble_vecteurs=[v1,v2,v3,v4]
```

Vérifiez que les vecteurs sont tous unitaires

$$\forall k \in [0, N[, \quad | \|v_k\| - 1 | < \epsilon$$

(On évitera la notation $v_k = a$ pour des nombres à virgule)

Application

```
v1=[1,0,0]
v2=[0,1,0]
v3=[1/(3**0.5),1/(3**0.5),1/(3**0.5)]
v4=[1/(2**0.5),0/(2**0.5),1/(2**0.5)]

ensemble_vecteurs=[v1,v2,v3,v4]
```

```
def norme(v):
    return sum(x_k**2 for x_k in v)**0.5

epsilon=1e-6
verification=all(abs(norme(v)-1)<epsilon for v in ensemble_vecteurs)

print(verification)
```

Cas d'application

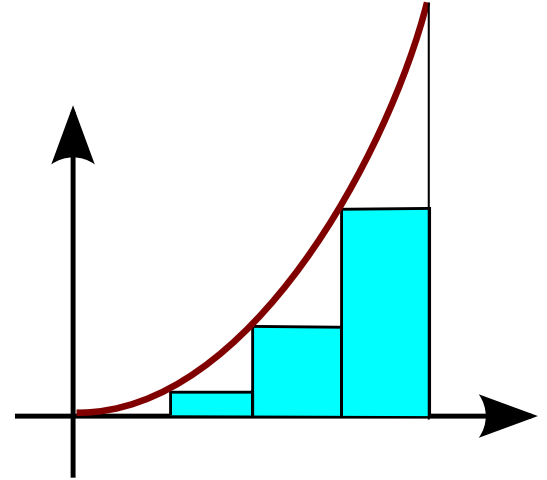
Calcul intégral

Calcul intégral

Soit $f(x) = ax^2 + bx + c$

$$\int_0^1 f(x) dx \simeq \sum_{k=0}^{k < N} f(k\Delta x) \Delta x$$

$$\Delta x = \frac{1}{N}$$

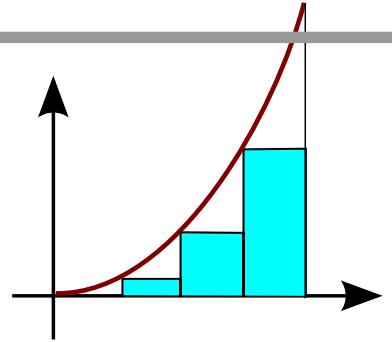


Calculer cette intégrale pour $a=3$, $b=2$, $c=1$ (prendre $N=100$)

Calcul intégral

$$\int_0^1 f(x) \, dx \simeq \sum_{k=0}^{N-1} f(k\Delta x) \Delta x \quad \Delta x = \frac{1}{N}$$

$$f(x) = ax^2 + bx + c$$



```
a=3
```

```
b=2
```

```
c=1
```

```
def f(x):
```

```
    return a*x**2+b*x+c
```

```
N=100
```

```
dx=1/N
```

```
I=dx*sum([f(k*dx) for k in range(0,N)])
```

```
print(I)
```

Calcul intégral

Quelle est l'erreur E par rapport à la vraie valeur?

Tracer le log de l'erreur en fonction de N

Calcul intégral

Quelle est l'erreur E par rapport à la vraie valeur?

Tracer le log de l'erreur en fonction de N

```
a=3
b=2
c=1

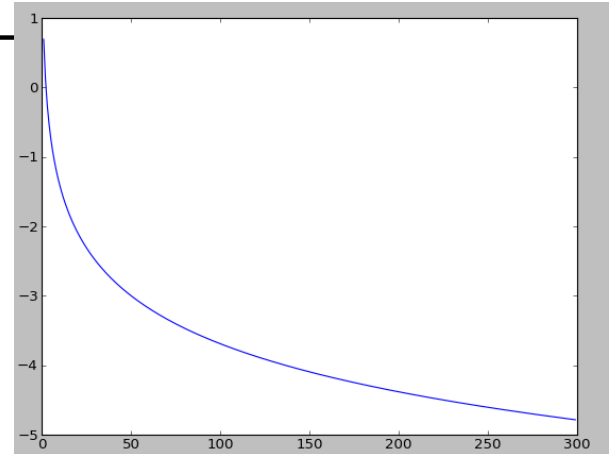
def f(x):
    return a*x**2+b*x+c

def vraie_integrale():
    return a/3+b/2+c

def integrale_numerique(N):
    dx=1/N
    return dx*sum([f(k*dx) for k in range(0,N)])

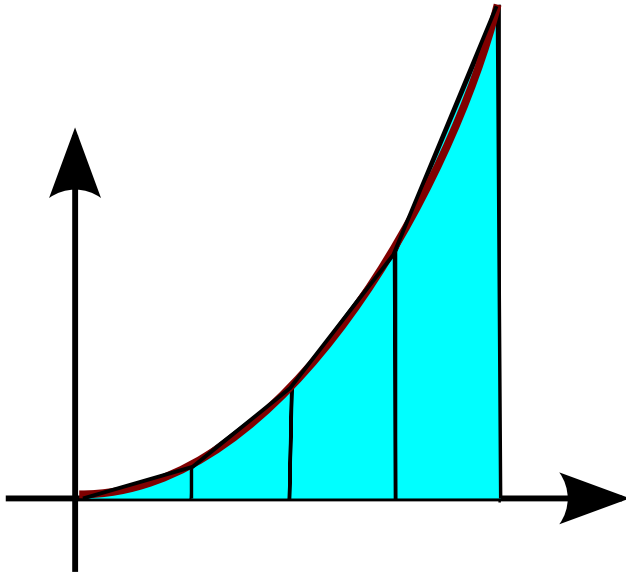
N_variable=range(1,300)
E=[log(abs(integrale_numerique(N)-vraie_integrale())) for N in N_variable]

plt.plot(N_variable,E)
plt.show()
```



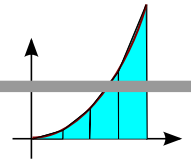
Calcul intégral

$$I = \int_0^1 f(x) \, dx \simeq \frac{\Delta x}{2} \sum_{k=0}^{k < N} f(k\Delta x) + f((k+1)\Delta x)$$



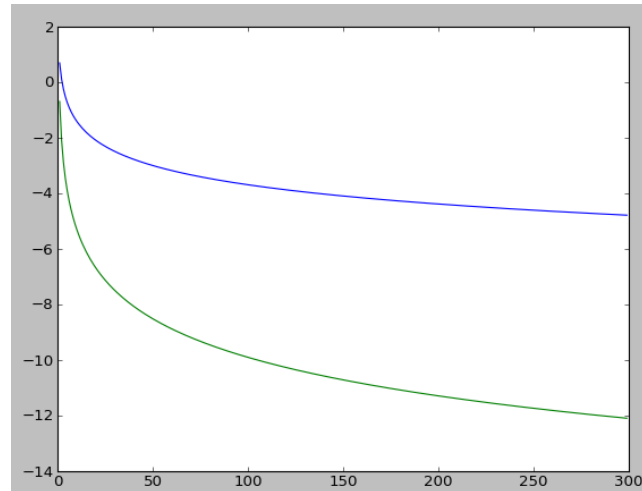
Meilleure approximation

Calcul intégral

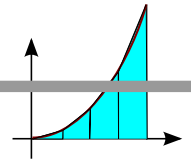


$$I = \int_0^1 f(x) \, dx \simeq \frac{\Delta x}{2} \sum_{k=0}^{k < N} f(k\Delta x) + f((k+1)\Delta x)$$

```
def integrale_numerique_trapeze(N):  
    dx=1/N  
    return dx/2*sum([f(k*dx)+f((k+1)*dx) for k in range(0,N)])
```



Calcul intégral



$$I = \int_0^1 f(x) \, dx \simeq \frac{\Delta x}{2} \sum_{k=0}^{k < N} f(k\Delta x) + f((k+1)\Delta x)$$

Rem, pour faire moins de calculs:

$$I = \frac{\Delta x}{2} \left(f(0) + f(1) + 2 \sum_{k=1}^{k < N} f(k\Delta x) \right)$$

```
def integrale_numerique_trapeze_2(N):  
    dx=1/N  
    return dx/2*(f(0)+f(1)+2*sum([f(k*dx) for k in range(1,N)]))
```

Calcul intégral

Evaluer la longueur de la courbe de f sur $[0,1]$

$$L = \int_0^1 (f'(x)^2 + 1)^{1/2} dx$$

Boucle for "avancée"

Boucle sur des mots

```
ensemble=["pommes","poires","champignons","poivrons"]  
[print("j'aime manger des "+aliment) for aliment in ensemble]
```

j'aime manger des pommes
j'aime manger des poires
j'aime manger des champignons
j'aime manger des poivrons

Boucle sur plusieurs vecteurs

```
ensemble_matiere=["math","physique","chimie","informatique"]  
ensemble_notes=[12.1,8.4,12.3,7.8]  
  
[print(matiere,":",note) for matiere , note  
in zip(ensemble_matiere,ensemble_notes)]
```

→ met les éléments ensemble

math : 12.1
physique : 8.4
chimie : 12.3
informatique : 7.8

Boucle "classique"

Remarque: Parfois/souvent f est

- complexe
- ne retourne rien / modifie x (la liste)
- n'est écrite qu'une seule fois

```
[ f(x) for x in nom_liste ]
```

```
for x in nom_liste:
```



```
f(x)
```

Boucle "classique"

Exemple

```
for x in range(0,8):  
    print(x)
```

```
for x in range(-3,4):  
    if(x%2 == 0):  
        print(x, "est pair")  
    else:  
        print(x, "est impair")
```

$a\%b$
reste de la division
euclidienne

Boucle "classique"

Soit:

```
ensemble_matières=["math","physique","chimie","informatique"]  
ensemble_notes=[9.1,8.4,16.3,7.8]
```

Afficher "attention + nom_matière" si note<10

Afficher "ATTENTION + nom_matière" si note<8

Afficher "TB + nom_matière" si note>15

Boucle "classique"

Soit:

```
ensemble_matiere=["math","physique","chimie","informatique"]  
ensemble_notes=[9.1,8.4,16.3,7.8]
```

Afficher "attention + nom_matiere" si note<10

Afficher "ATTENTION + nom_matiere" si note<8

Afficher "TB + nom_matiere" si note>15

```
for matiere,note in zip(ensemble_matiere,ensemble_notes):  
    if(note<8):  
        print("ATTENTION",matiere,"!!!")  
    elif(note<10):  
        print("attention",matiere)  
    elif(note>15):  
        print("TB",matiere)
```

Boucle "classique"

Modification d'éléments:

Ajouter 2 à toutes les notes si elles sont inférieures à 8

```
notes=[9.1,8.4,16.3,7.8]
for k in range(len(notes)):
    if(notes[k]<8):
        notes[k]=notes[k]+2
```

Récupérer valeur et indice

```
ma_liste=[9.1,8.4,16.3,7.8]  
for indice,valeur in enumerate(ma_liste):  
    print(indice,valeur)
```

0	9.1
1	8.4
2	16.3
3	7.8

Similaire à :

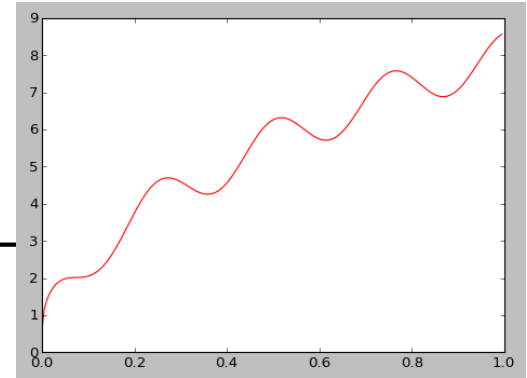
```
for indice in range(len(ma_liste)):  
    valeur=ma_liste[indice]  
  
    print(indice,valeur)
```

Récupérer valeur et indice

Application: Rendre une suite croissante

Soit la suite $(a_k)_{k \in \llbracket 0, N \rrbracket}$

Si $a_{k+1} < a_k$, alors $a_{k+1} := a_k$



$$\forall k \in \llbracket 0, N \rrbracket, a_k := f(k/N)$$

$$\forall x \in [0, 1], f(x) := 8\sqrt{x} + 0.6 \cos(25x)$$

Récupérer valeur et indice

Application: Rendre une suite croissante

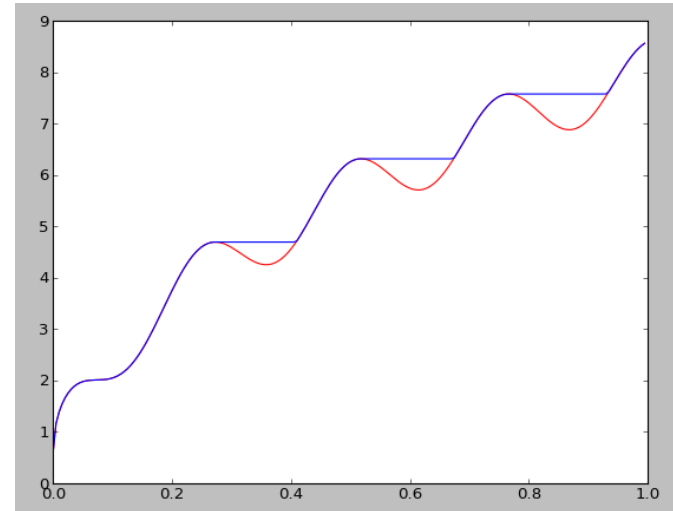
Soit la suite $(a_k)_{k \in \llbracket 0, N \rrbracket}$

Si $a_{k+1} < a_k$, alors $a_{k+1} := a_k$

$\forall k \in \llbracket 0, N \rrbracket, a_k := f(k/N)$

$\forall x \in [0, 1], f(x) := 8\sqrt{x} + 0.6 \cos(25x)$

```
for k, a_k in enumerate(a):  
    if k+1 < N and a[k+1] < a[k]:  
        a[k+1] = a[k]
```



Application

Soit:

```
matieres=["math","physique","chimie","informatique"]  
notes=[9.1,8.4,11.3,6.8]
```

Soit m la moyenne des notes

Ajouter 2 points à chaque note si m est inférieur à 10

Ajouter seulement 1 point pour les maths

Si m inférieur à 8, rajouter 3 points à la chimie

Application

Soit:

```
matieres=["math","physique","chimie","informatique"]  
notes=[9.1,8.4,11.3,6.8]
```

Soit m la moyenne des notes

Ajouter 2 points à chaque note si m est inférieur à 10

Ajouter seulement 1 point pour les maths

Si m inférieur à 8, rajouter 3 points à la chimie

```
matieres=["math","physique","chimie","informatique"]  
notes=[9.1,8.4,11.3,6.8]  
  
moyenne=sum(notes)/len(notes)  
  
if(moyenne<10):  
    for k in range(len(notes)):  
        if matiere[k]=="math":  
            notes[k] = notes[k]+1  
        elif matiere=="chimie" and moyenne<8:  
            notes[k] = notes[k]+3  
        else:  
            notes[k] = notes[k]+2  
  
[print(matiere,":",note) for matiere , note  
 in zip(matieres,notes)]
```

Application

Soit $f(x) = E(x) \% 2$ E : partie entière ($\text{int}(x)$)
 x appartenant à $[0, 10]$

Calculer $N=200$ échantillons (y_k) de f sur $[0, 10[$

Calculer z_k , tel que $z_k = 1/3 (y_{k+1} + y_k + y_{k-1})$
pour k dans $[1, N-1[$, sinon $z_k = y_k$ pour $k=0$ et $k=N$

Itérer le processus sur z ... 15 fois

$$g(x) = \int_{y=x-\epsilon}^{y=x+\epsilon} f(y) dy$$

Application

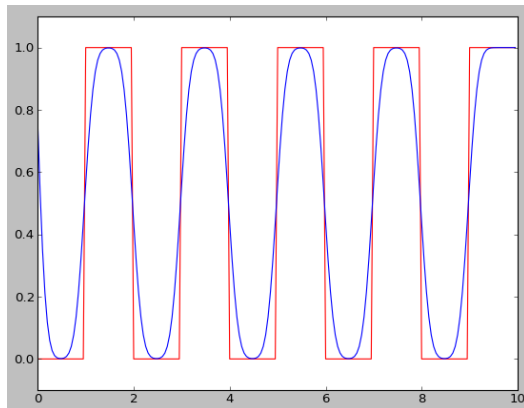
Soit $f(x) = E(x) \% 2$ E : partie entière ($\text{int}(x)$)
 x appartenant à $[0,10]$

Calculer $N=200$ échantillons (y_k) de f sur $[0,10[$

Calculer z_k , tel que $z_k = 1/3 (y_{k+1} + y_k + y_{k-1})$
pour k dans $[1, N-1[$, sinon $z_k = y_k$ pour $k=0$ et $k=N$

Itérer le processus sur z ... 15 fois

$$g(x) = \int_{y=x-\epsilon}^{y=x+\epsilon} f(y) dy$$

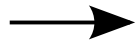


```
def f(x):  
    return int(x)%2  
  
def g(y):  
  
    y2=list(y);  
  
    for k,y_k in enumerate(y[1:-1]):  
        y2[k]=1/3*(y[k-1]+y[k]+y[k+1])  
  
    return y2  
  
N=200;  
x=[10*k/N for k in range(N)]  
y=[f(x_k) for x_k in x]  
  
plt.plot(x,y,"r")  
  
for k in range(15):  
    y=g(y)  
  
plt.plot(x,y)  
plt.axis([0,10,-0.1,1.1])  
plt.show()
```

Boucle while / "tant que"

Boucle while / "tant que"

while *condition_vraie* :



Faire quelque chose

```
a=5
while a>2:
    a=0.65*(a+1)
    print(a)
```

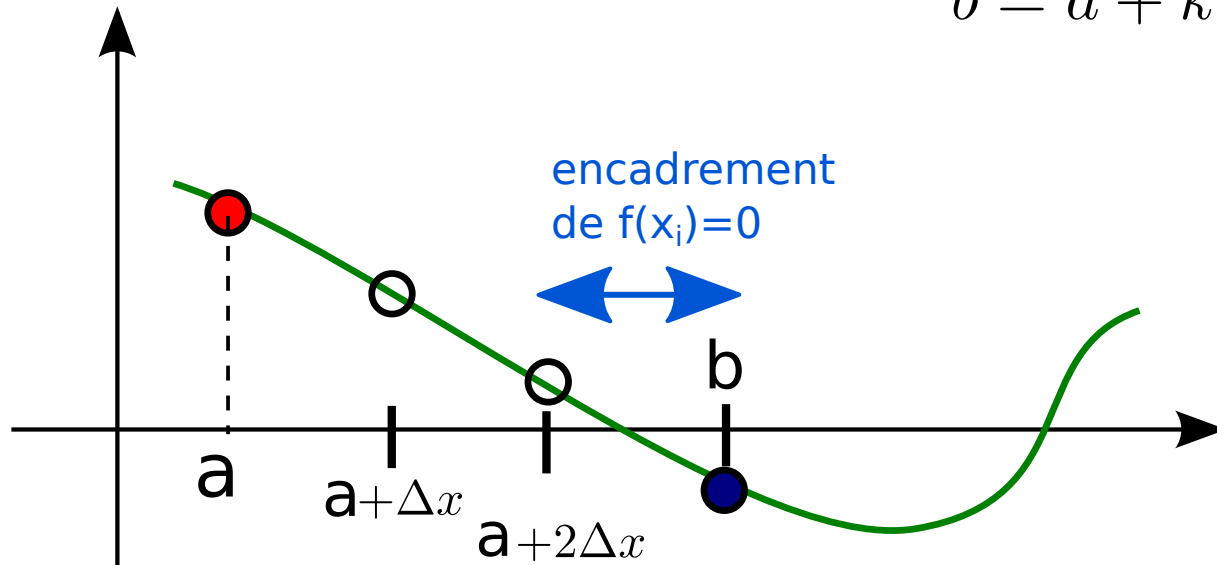
On ne connaît pas forcément
le nombre d'itérations

Application: encadrement

Soit f une fonction continue de $\mathbb{R} \rightarrow \mathbb{R}$

Soit a , $f(a) > 0$ Trouver b , $f(b) < 0$

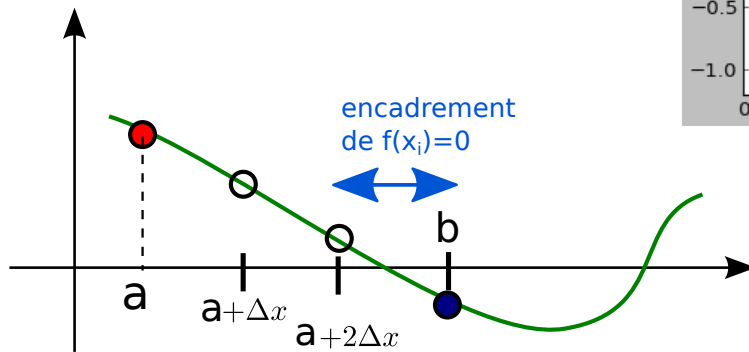
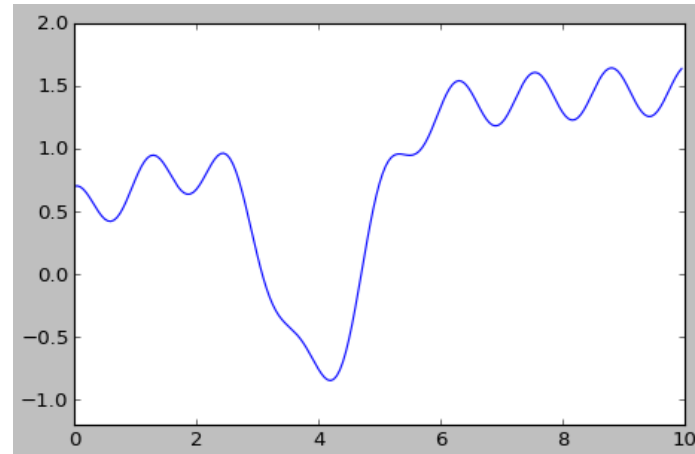
$$b = a + k\Delta x$$



Application: encadrement

$$f(x) = \frac{1}{2} + \tanh\left(\frac{x}{5}\right) - 2e^{-(x-4)^2} + 0.2\cos(5x)$$

$$a = 0 \quad \Delta x = 0.1$$



Application: encadrement

$$f(x) = \frac{1}{2} + \tanh\left(\frac{x}{5}\right) - 2e^{-(x-4)^2} + 0.2 \cos(5x)$$

$$a = 0 \quad \Delta x = 0.1$$

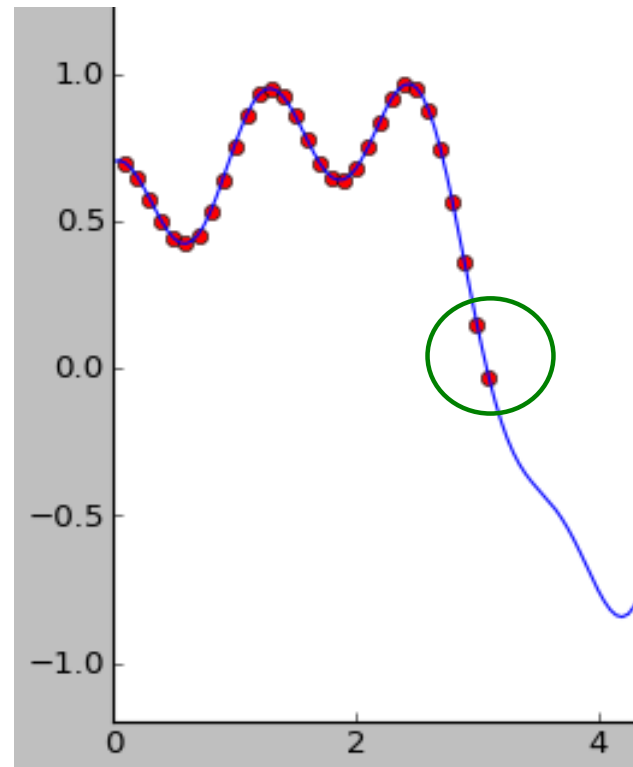
`dx=0.1`

`a=0`

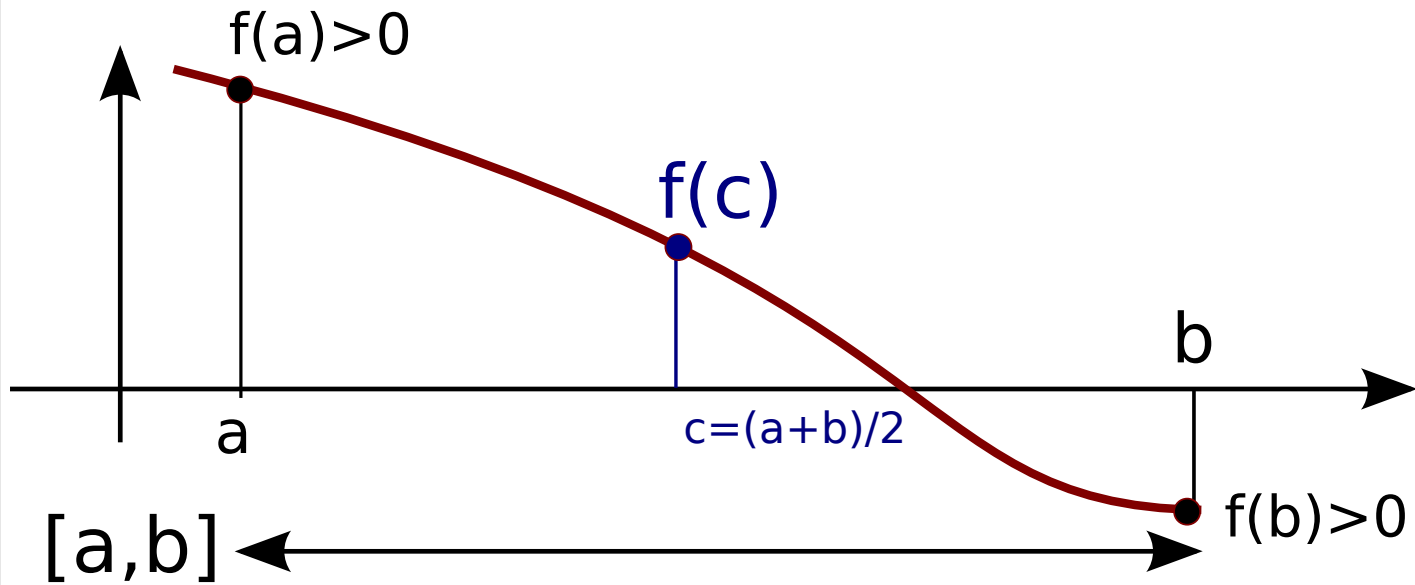
`x=a`

`while f(x)>0:`

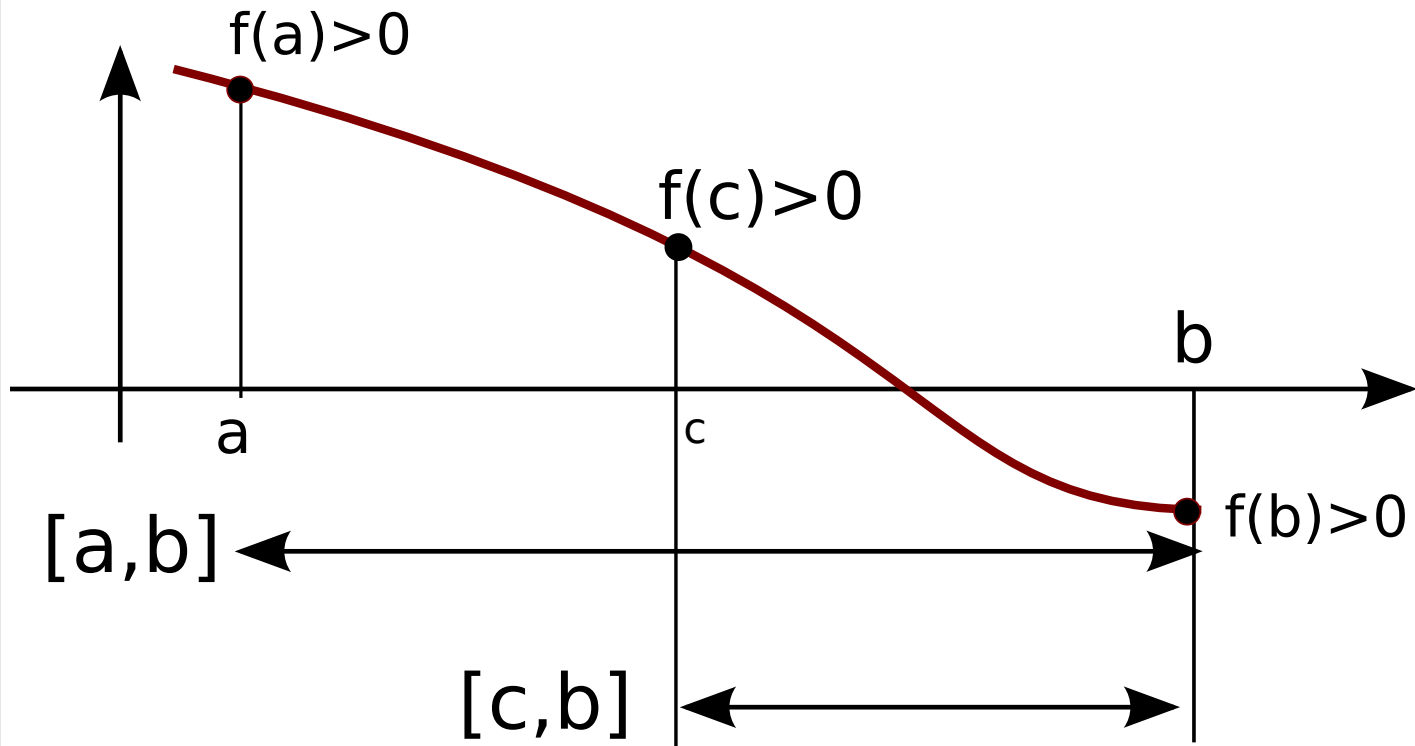
`x=x+dx`



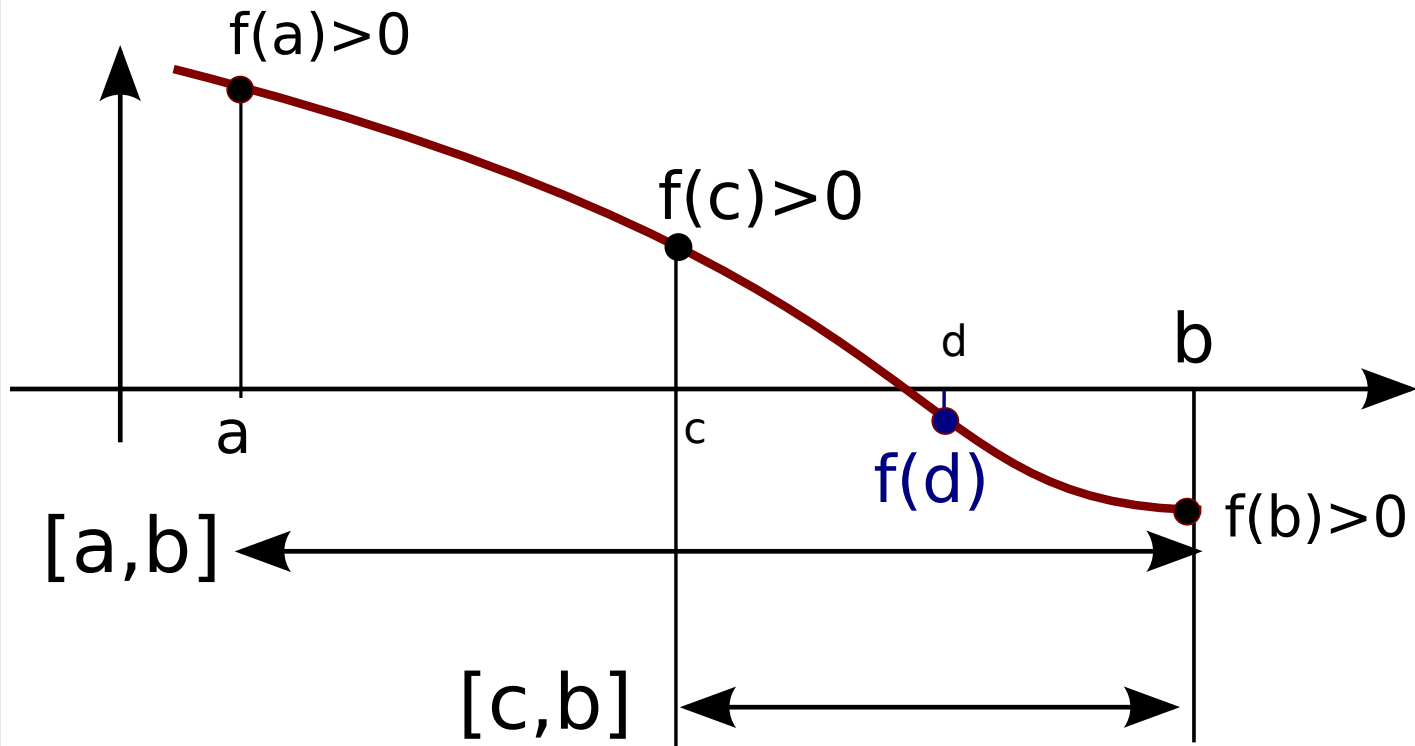
Dichotomie



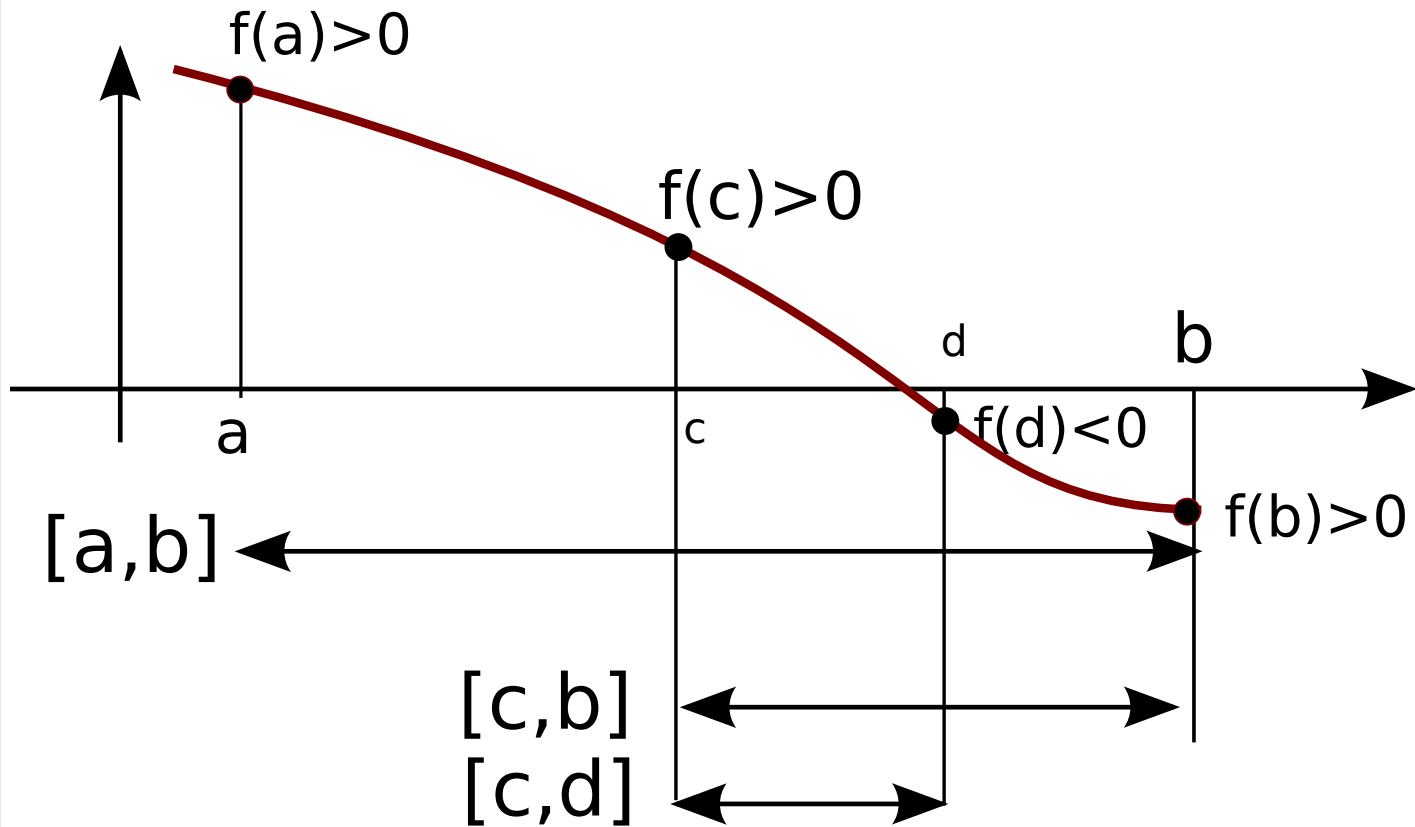
Dichotomie



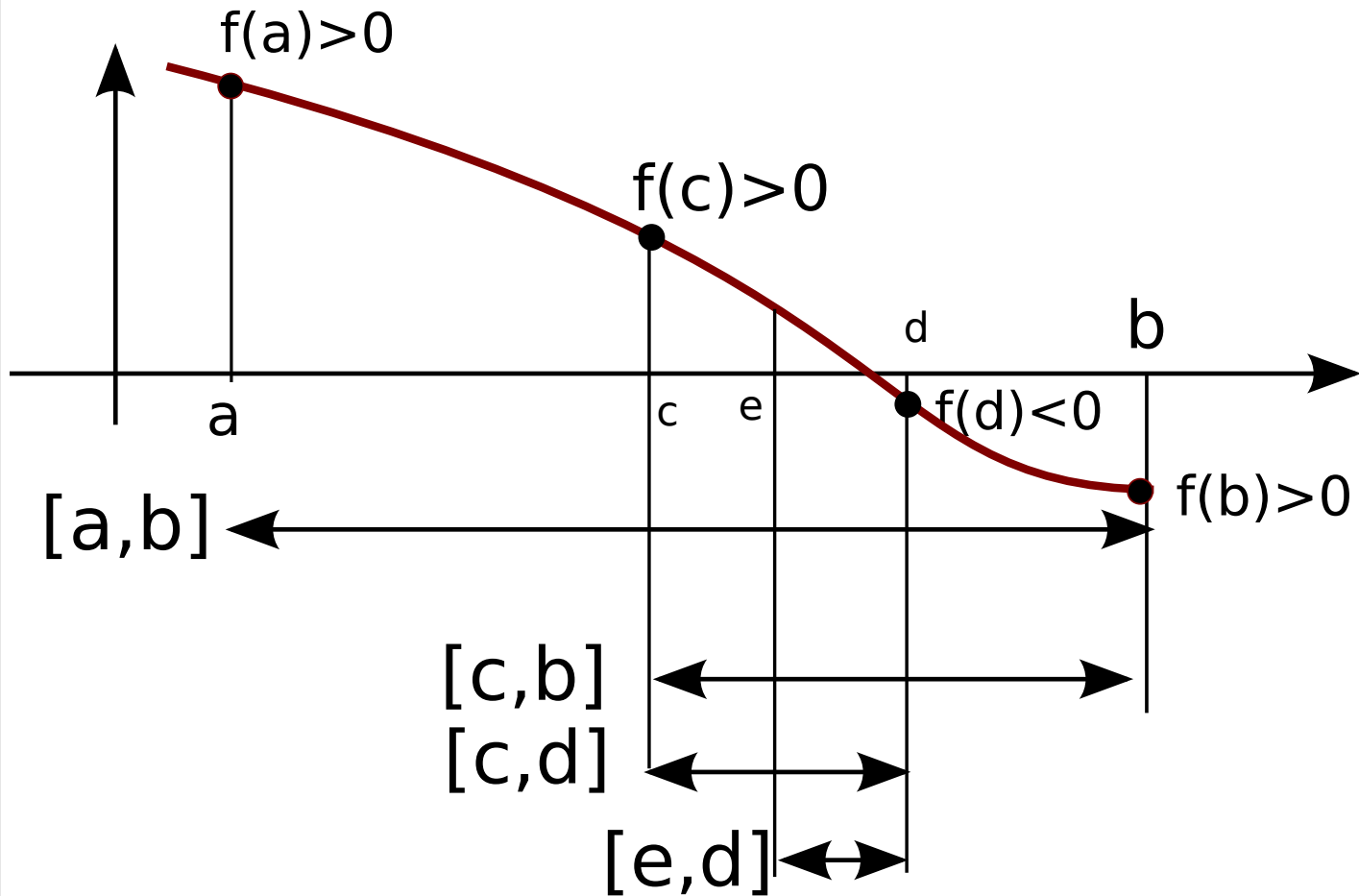
Dichotomie



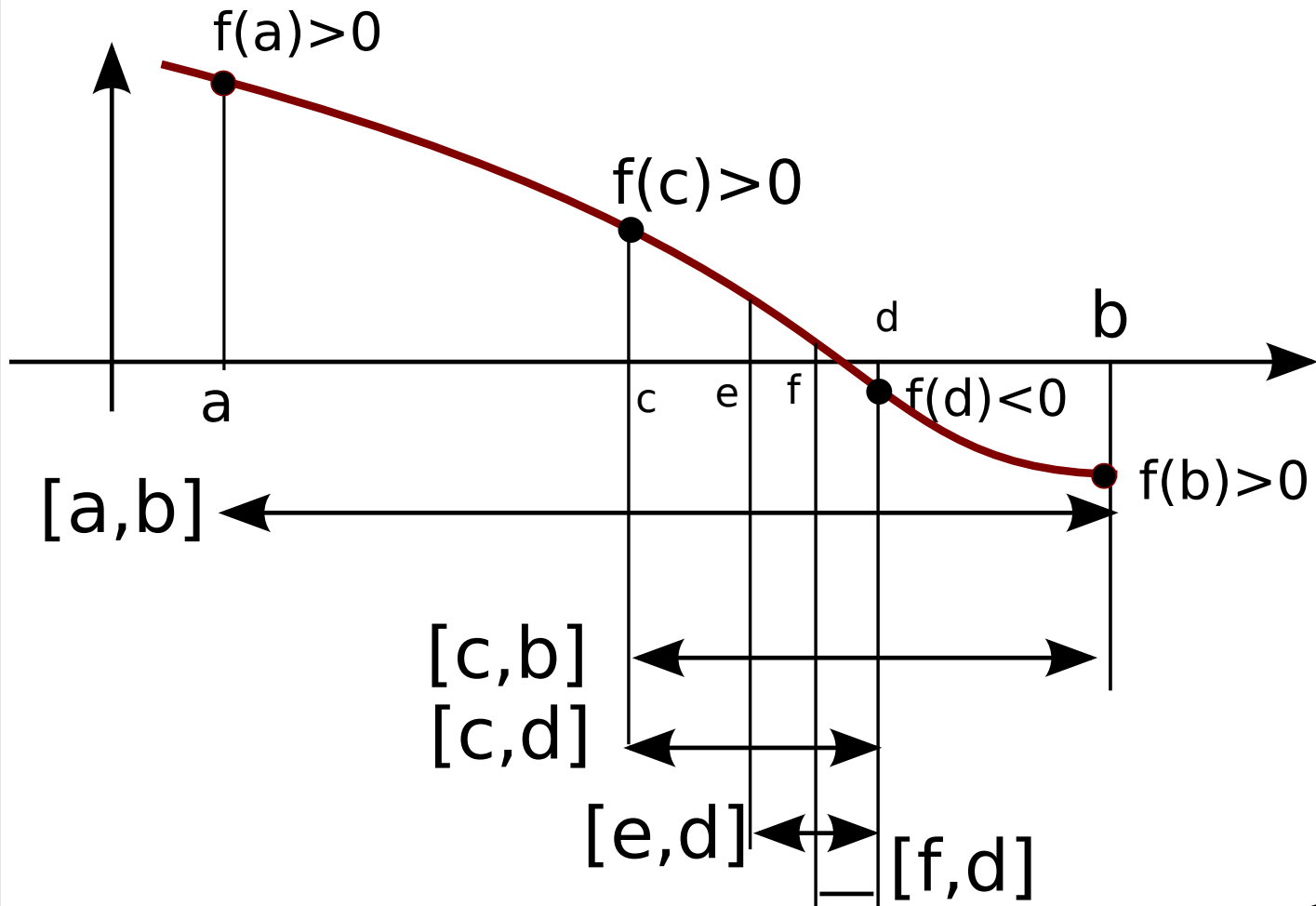
Dichotomie



Dichotomie



Dichotomie



Dichotomie

Algorithme

Soit $[a,b]$, $f(a)>0$ et $f(b)<0$

Tant que $|b-a|>\text{erreur_max}$

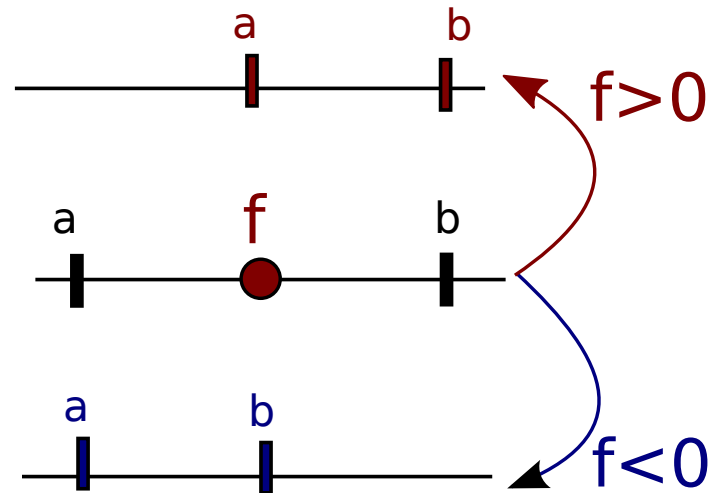
 Calculer $f(c)$, avec $c=(a+b)/2$

Si $f(c)>0$

$[a,b] \leftarrow [c,b]$

Sinon

$[a,b] \leftarrow [a,c]$



Dichotomie

Application: calculer le zéro de f
pour un intervalle d'encadrement de 10^{-6}

Dichotomie

Application: calculer le zéro de f
pour un intervalle d'encadrement de 10^{-6}

```
dx=0.1
a=0
x=a
while f(x)>0:
    x=x+dx
a=x-dx
b=x

epsilon=1e-6
while abs(b-a)>epsilon:
    c=(a+b)/2
    if f(c)>0:
        a=c
    else:
        b=c

print(c, f(a), f(b))
```

Librairie mathématique et affichage

Numpy: Array

```
import numpy as np

va=np.array([1,4,8,9])
vb=np.array([-4.1,2.2,1,-5])

vc=va+vb

print(vc)
```

array ressemble aux listes
spécialisé pour les nombres

Numpy: Array

```
va=np.array([1,4,8,9])  
vb=np.array([-4.1,2.2,1,-5])
```

```
va+vb  
va-vb  
2*va  
vb*4  
np.vdot(va,vb)
```

addition
soustraction
multiplication par un scalaire
produit scalaire
...

Rem. On ne mélange pas "mot" et nombre dans un array

Linspace

```
a,b=1.1,4.8  
N=8  
  
x=np.linspace(a,b,N)  
  
print(x)
```

Vecteur uniformément réparti entre $[a,b]$ avec N échantillons

Affichage

Combinaison array + affichage

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

```
a,b=-4,4
```

```
N=200
```

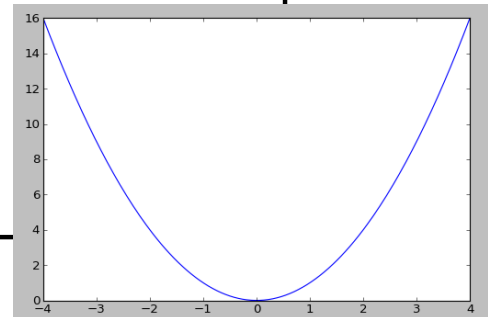
```
x=np.linspace(a,b,N)
```

```
y=x**2
```

```
plt.plot(x,y)
```

```
plt.show()
```

élève au carré
élément à élément



Array-range

arange: similaire à range

```
v=np.arange(1,8,2)  
print(v)
```

```
1 3 5 7
```

Slicing

```
print(a)  
print(a[2:4])  
print(a[2:])  
print(a[:5])  
print(a[1:6:2])  
print(a[::-3])
```

Vecteurs particuliers

```
va=np.zeros(5)  
vb=np.ones(6)
```

va

0	0	0	0	0
---	---	---	---	---

vb

1	1	1	1	1	1
---	---	---	---	---	---

Applications

Soit la droite D passant par x_0 et de vecteur directeur u

$$x_0 = (1, 2)$$

$$u = (1, 4)$$

Calculer u_n , le vecteur directeur unitaire de même direction que u
(*norme: from numpy.linalg import norm*)

Soit $a = x_0 - 2u_n$, $b = x_0 + 2u_n$

Afficher la droite ab

Afficher un point en x_0

Applications

Soit $p=(3,3)$

Calculer $L=\langle ap, un \rangle$

Calculer la projection orthogonale p_{proj} de p sur la droite D
 $p_{proj}=a+L un$

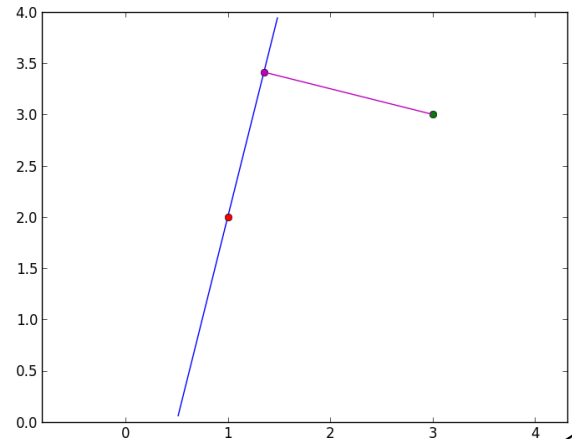
Afficher p et p_{proj}

Notez que les axes ne sont pas orthogonaux

Ajouter: `plt.axis("square")`

Afficher le segment $[p, p_{proj}]$

Calculer la distance entre p et la droite D



Applications

Soit C le cercle de centre $x_0=(1,2)$ et de rayon 4

Construire le vecteur θ contenant N échantillons
également répartis sur $[0, 2\pi]$

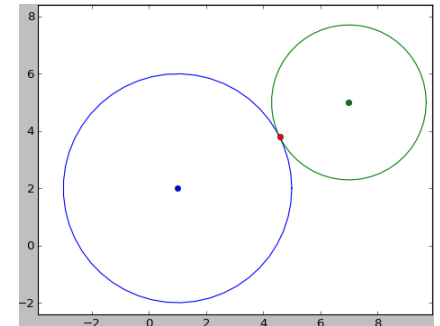
($\pi = \text{np.pi}$)

Stocker dans les vecteurs cx , et cy les coordonnées
de N échantillons du cercle C

Tracer le cercle C

Soit C' un cercle tangent à C de centre $x_0'=(7,5)$

Calculer le rayon de C'



Tracer C' , les points x_0 et x_1 et le point d'intersection entre C et C'

Applications

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import norm

x0=np.array([1,2])
r=4

N=50
theta=np.linspace(0,2*np.pi,N)

cx=r*np.cos(theta)+x0[0]
cy=r*np.sin(theta)+x0[1]

x1=np.array([7,5])
r1=norm(x0-x1)-r

c1x=r1*np.cos(theta)+x1[0]
c1y=r1*np.sin(theta)+x1[1]

inter=x0+(x1-x0)/norm(x1-x0)*r

plt.plot(cx,cy)
plt.plot(c1x,c1y,"g")
plt.plot(x0[0],x0[1],"bo")
plt.plot(x1[0],x1[1],"go")
plt.plot(inter[0],inter[1],"ro")

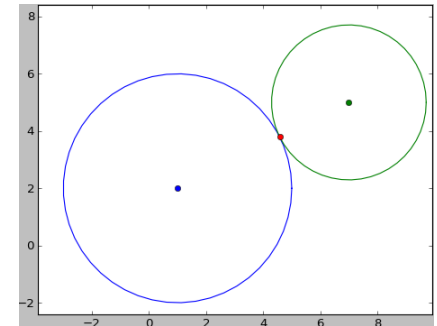
plt.axis('equal')
plt.show()
```

de $x_0=(1,2)$ et de rayon 4

theta contenant N échantillons
dans $[0, 2\pi]$ ($\pi=np.\pi$)

pour c_x , et c_y les coordonnées
du cercle C

et à C' de centre $x_0'=(7,5)$



Tracer C', les points x_0 et x_1 et le point d'intersection entre C et C'

Vecteurs multi-dimensionnels

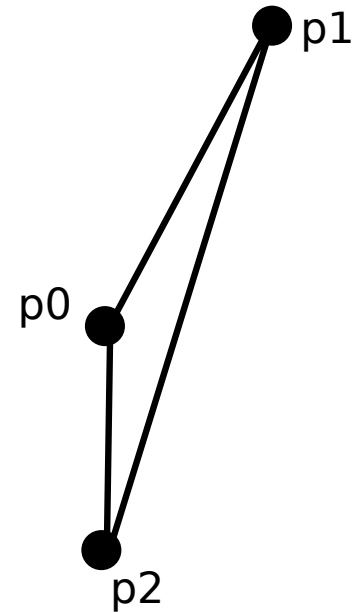
```
p0=np.array([1,2])
p1=np.array([4,7])
p2=np.array([1,-2])

triangle=np.array([p0,p1,p2])

print(triangle[2,1]) #coord-y p2

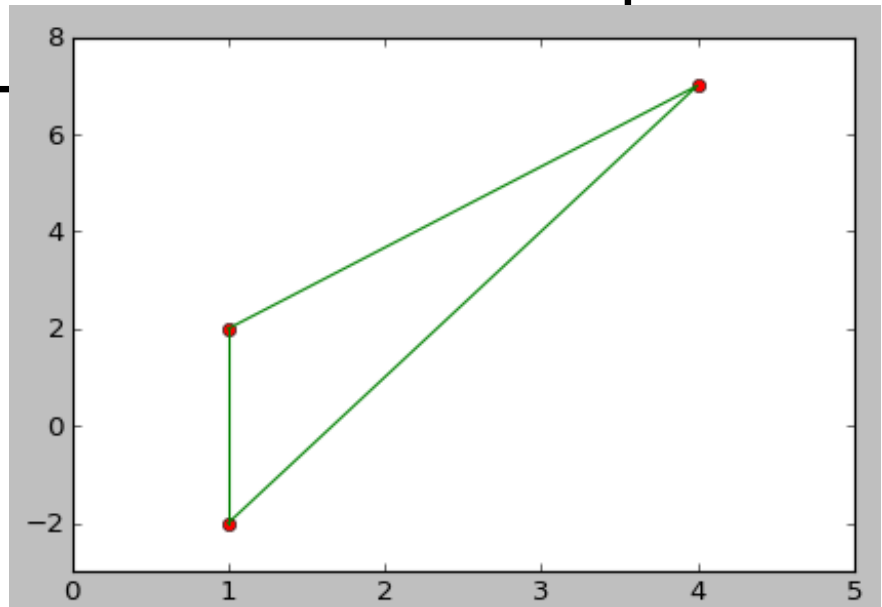
print(triangle[0,:]) #point 0
print(triangle[1,:]) #point 1
print(triangle[2,:]) #point 2

print(triangle[:,0]) #coord-x
print(triangle[:,1]) #coord-y
```



Vecteurs multi-dimensionnels

```
plt.plot(triangle[:,0],triangle[:,1],"ro")  
plt.plot(triangle[:,0],triangle[:,1],"g-")  
plt.plot([triangle[0,0],triangle[2,0]],  
         >> [triangle[0,1],triangle[2,1]],"g-")  
plt.axis([0,5,-3,8])  
plt.show()
```



Embellissement graphique

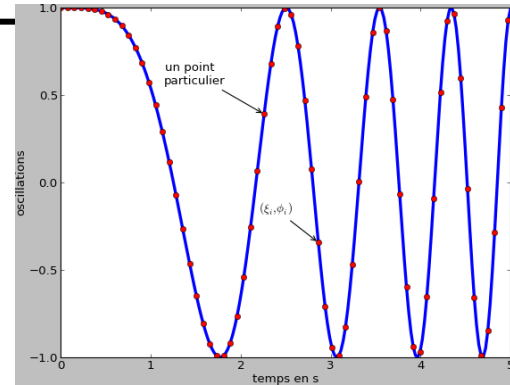
```
N=200
x=np.linspace(0,5,N)
y=np.cos(x*x)

plt.plot(x,y,linewidth=3)
plt.plot(x[::3],y[::3],"ro")
plt.xlabel("temps en s")
plt.ylabel("oscillations")

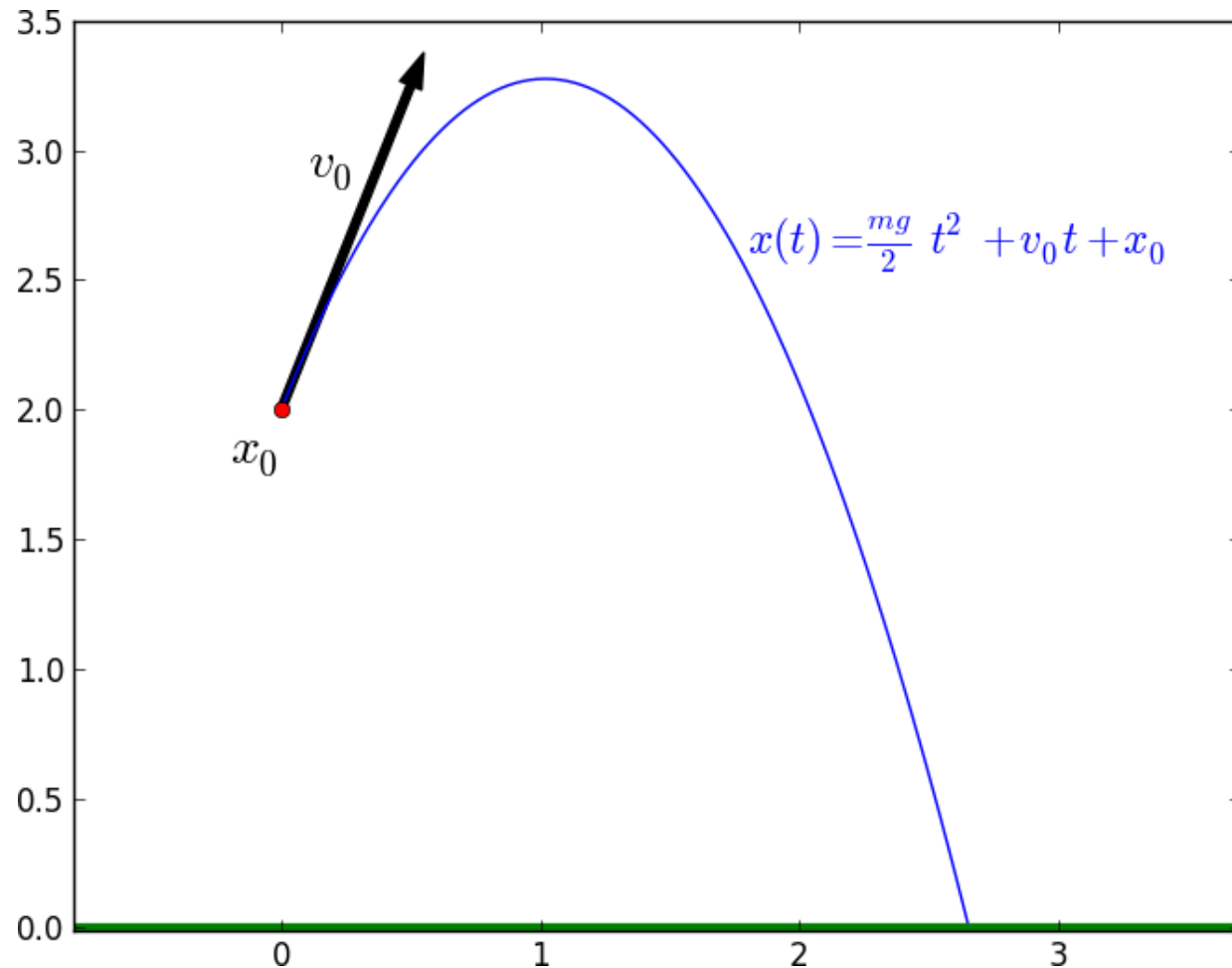
plt.annotate('un point \nparticulier', xy=(x[90], y[90]), xycoords='data',
            xytext=(-100, 30), textcoords='offset points',
            arrowprops=dict(arrowstyle="->"))

plt.annotate(u'$\\xi_i, \\phi_i$', xy=(x[114], y[114]), xycoords='data',
            xytext=(-60, 30), textcoords='offset points',
            arrowprops=dict(arrowstyle="->"))

plt.show()
```



Application, afficher:



Application, afficher:

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import norm
from mpl_toolkits.mplot3d import Axes3D
```

```
x0=np.array([0,2])
v0=np.array([2,5])
g=np.array([0,-9.8])
```

```
N=100
```

```
t=np.linspace(0,1.4,N)
```

```
x=np.array([np.zeros(N),np.zeros(N)]);
```

```
for dim in range(2):
    x[dim]=1/2*g[dim]*t**2 + v0[dim]*t+x0[dim]
```

```
#plt.arrow(0,0,0.5,0.5,width=0.1)
```

```
plt.arrow(x0[0],x0[1],v0[0]/4,v0[1]/4,width=0.03,color="black")
```

```
plt.plot(x[0,:],x[1:],linewidth=1)
```

```
plt.plot([-5,5],[0,0],"g-",linewidth=3)
```

```
plt.plot(x0[0],x0[1],"ro")
```

```
plt.text(x0[0]-0.2,x0[1]-0.2,u"$x_0$",fontsize=20)
```

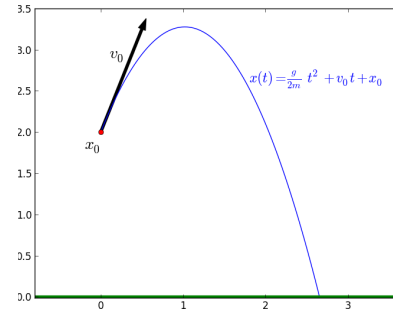
```
plt.text(x0[0]+0.1,x0[1]+0.9,u"$v_0$",fontsize=20)
```

```
plt.text(1.8,2.6,u"$x(t)=\\frac{g}{2m}t^2+v_0t+x_0$",fontsize=17,color="blue")
```

```
plt.axis('equal')
```

```
plt.axis([-0.1,3,-0.01,3.5])
```

```
plt.show()
```



Courbe et points 3D

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

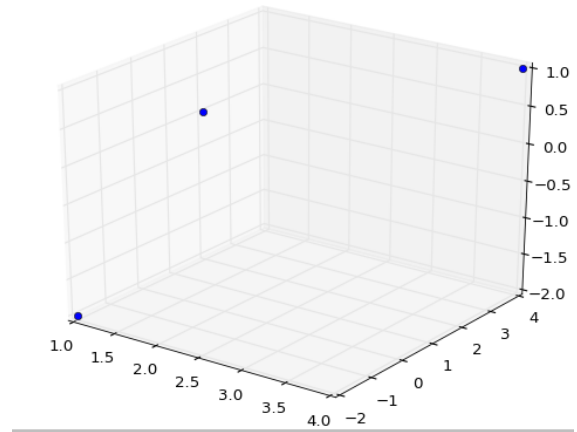
p0=np.array([1,2,0])
p1=np.array([4,4,1])
p2=np.array([1,-2,-2])

triangle=np.array([p0,p1,p2])

x=triangle[:,0]
y=triangle[:,1]
z=triangle[:,2]

print(x,y,z)

fig=plt.figure()
axes3d=fig.gca(projection='3d')
axes3d.plot(triangle[:,0],triangle[:,1],triangle[:,2],"o")
plt.show()
```



Courbe et points 3D

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
N=200
```

```
t=np.linspace(0,5*np.pi,N)
```

```
x=(1-t/5)*np.cos(2*t)
```

```
y=(1-t/5)*np.sin(2*t)
```

```
z=t/2
```

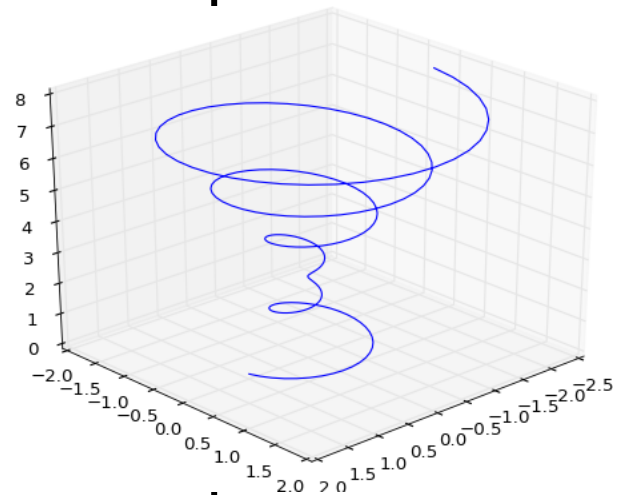
```
p=np.array([x,y,z])
```

```
fig=plt.figure()
```

```
axes3d=fig.gca(projection='3d')
```

```
axes3d.plot(p[0,:],p[1,:],p[2:], "-")
```

```
plt.show()
```



Cas d'application: Equations différentielles

Dérivée discrète

Soit une fonction réelle dérivable f

La dérivée discrète de f au point x est calculable par la relation

$$\frac{f(x + \epsilon) - f(x)}{\epsilon}$$

Application:

Coder la fonction: $f(x)=\sin(x)$

Calculer la dérivée numérique en 0

Comparer à la vraie valeur (pour différentes valeurs de ϵ)

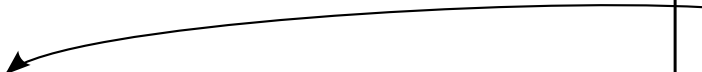
Dérivée discrète

Rem. On peut définir l'application

$$(\mathcal{F} \times \mathbb{R}) \rightarrow \mathbb{R}$$
$$D : (f, x) \mapsto f'(x)$$

```
def f(x):  
    return np.sin(x)  
  
def D(f,x):  
    epsilon=1e-6  
    df=(f(x+epsilon)-f(x))/epsilon  
  
    return df  
  
x=0  
print(D(f,0))  
print(D(f,0.5))  
print(D(f,0.8))
```

f est un
argument de D



Dérivée discrète

Application:

$$g(x) = \tanh(\sin(x) + \tanh(x))$$

$$f(x) = (g \circ g \circ g)(x)$$

Afficher f et f' sur $[-30, 30]$

Dérivée discrète

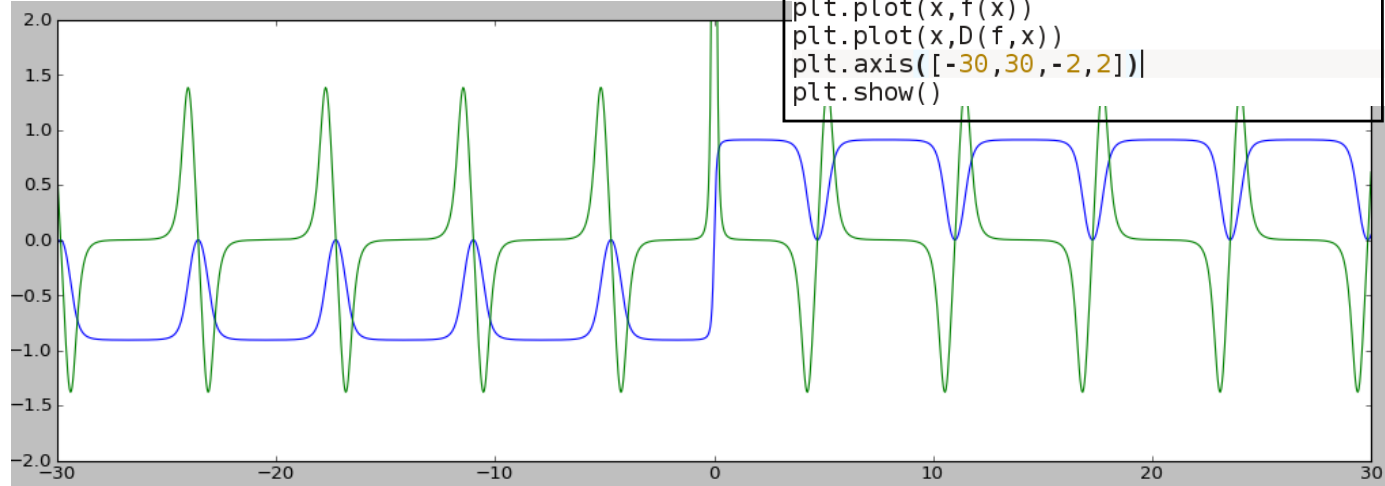
Application:

$$g(x) = \tanh(\sin(x) + \tanh(x))$$

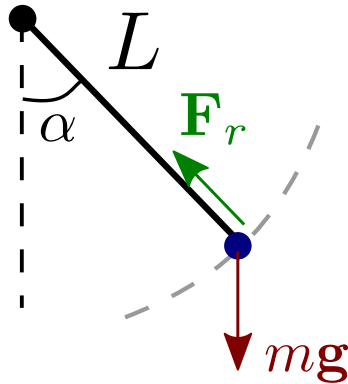
$$f(x) = (g \circ g \circ g)(x)$$

Afficher f et f' sur [-30,30]

```
def g(x):  
    return np.tanh(np.sin(x)+np.tanh(x))  
  
def f(x):  
    y=x  
    for k in range(3):  
        y=g(y)  
  
    return y  
  
def D(f,x):  
    epsilon=1e-6  
    df=(f(x+epsilon)-f(x))/epsilon  
  
    return df  
  
N=1500  
x=np.linspace(-30,30,N)  
  
plt.plot(x,f(x))  
plt.plot(x,D(f,x))  
plt.axis([-30,30,-2,2])  
plt.show()
```



Pendule oscillant



$$\alpha'(t) = \omega(t)$$

Equation de la dynamique:

$$\omega'(t) = -\frac{g}{L} \sin(\alpha(t))$$

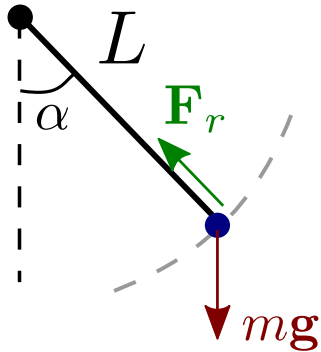
Pas de solution analytique simple pour $\alpha(t)$ grand

Si frottements, ou pendule couplé, pas de solution analytique du tout

On discrétise la dérivée

(on intègre numériquement suivant t)

Pendule oscillant



$$\alpha'(t) = \omega(t)$$

Equation de la dynamique:

$$\omega'(t) = -\frac{g}{L} \sin(\alpha(t))$$

Equation discrétisée:

$$\left\{ \begin{array}{l} \omega^{k+1} = \omega^k - \Delta t \frac{g}{L} \sin(\alpha^k) \\ \alpha^{k+1} = \alpha^k + \Delta t \omega^{k+1} \\ \alpha^0 = \alpha_0 \\ \omega^0 = \omega_0 \end{array} \right. \quad \text{conservation d'énergie}$$

Afficher $\alpha(t)$ en fonction de t

Considérer $\alpha_0 \simeq \pi$

Pendule oscillant

Algorithme:

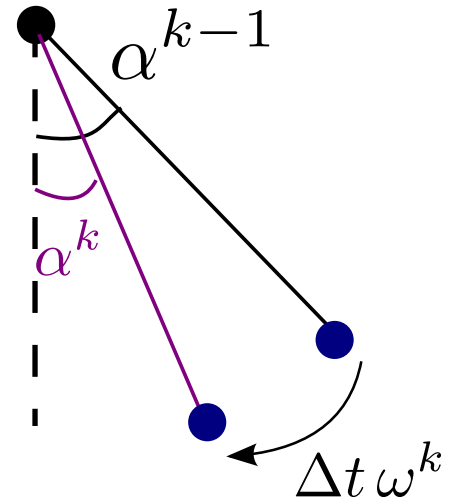
Initialiser α^0 , $\omega^0 \leftarrow 0$

Pour tous k

$$\omega^k = \omega^{k-1} - \Delta t \frac{g}{L} \sin(\alpha^{k-1})$$

$$\alpha^k = \alpha^{k-1} + \Delta t \omega^k$$

Afficher($t = k \Delta t$, α^k)



Pendule oscillant

Algorithme:

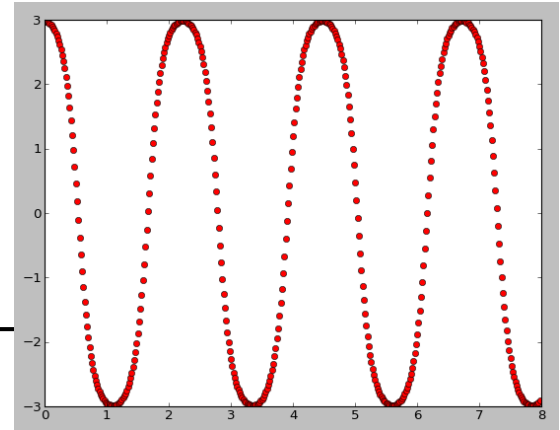
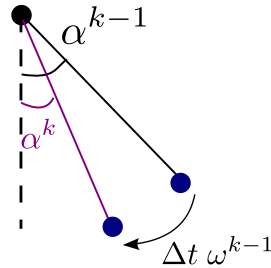
Initialiser α^0, ω^0

Pour tous k

$$\omega^k = \omega^{k-1} - \Delta t \frac{g}{L} \sin(\alpha^{k-1})$$

$$\alpha^k = \alpha_{k-1} + \Delta t \omega^{k-1}$$

Afficher($t = k \Delta t, \alpha^k$)



```
g=9.8
L=0.2
dt=0.02

omega=0
alpha=0.6*np.pi

omega_previous=omega
alpha_previous=alpha

for k in range(900):
    omega=omega_previous-dt*g/L*np.sin(alpha_previous)
    alpha=alpha_previous+omega*dt

    plt.plot(k*dt,alpha,"ro")

    omega_previous=omega
    alpha_previous=alpha

plt.show()
```

Matrices

Matrices

```
import numpy as np  
  
A=np.matrix([[1,2,3],[4,5,6]])  
  
print(A)
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Matrices

Produit matriciel

```
import numpy as np

A=np.matrix([[1,2,3],
             [4,5,6]])

B=np.matrix([[1,4],
             [1,2],
             [3,10]])

C=A*B

print(C)
```

Matrices

Matrices carrées

```
A=np.matrix([[1,2,3],  
              [4,5,6],  
              [7,8,9]])  
  
B=np.matrix([[1,4,2],  
              [1,2,2],  
              [3,-1,1]])  
  
print(A+B)  
print(A-B)  
print(A*B)  
print(A**2)
```

Matrices

Bloc/slicing

```
A=np.matrix([[1,2,3],  
             [4,5,6],  
             [7,8,9]])
```

```
A[0:2,1:3]
```

```
A[1:3,:]
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Operateurs matriciels

```
A=np.matrix([[1,2],  
              [4,5]])
```

```
np.linalg.det(A)
```

```
np.trace(A)
```

```
np.linalg.inv(A)
```

linalg = linear algebra

Algèbre linéaire

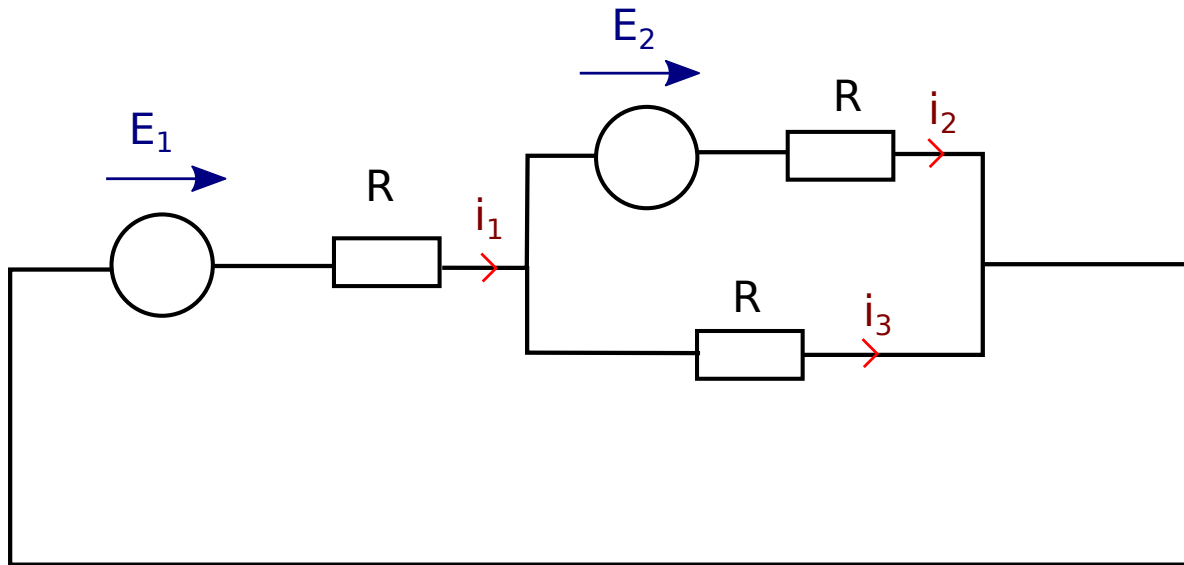
```
A=np.matrix([[1,2,5],  
              [4,5,9],  
              [7,8,4]])
```

```
b=np.array([4,8,9])
```

```
x=np.linalg.solve(A,b)
```

$$A x = b$$

Application, systeme lineaire

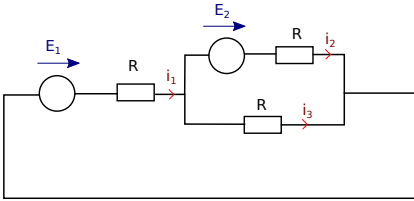


$$\begin{cases} -E_1 + Ri_1 - E_2 + Ri_2 = 0 \\ -E_1 + Ri_1 + Ri_3 = 0 \\ i_1 = i_2 + i_3 \end{cases}$$

$$\begin{cases} R = 10k\Omega \\ E_1 = 1V \\ E_2 = 0.5V \end{cases}$$

Calculer i_1 , i_2 , i_3

Application, systèmes lineaires



$$\begin{cases} -E_1 + Ri_1 - E_2 + Ri_2 = 0 \\ -E_1 + Ri_1 + Ri_3 = 0 \\ i_1 = i_2 + i_3 \end{cases} \quad \begin{cases} R = 10k\Omega \\ E_1 = 1V \\ E_2 = 0.5V \end{cases}$$

Calculer i_1 , i_2 , i_3

```
A=np.matrix([[R,R,0],  
              [R,0,R],  
              [1,-1,-1]])
```

```
e=np.array([E1+E2,E1,0])
```

```
i=np.linalg.solve(A,e)
```

```
print(i)
```

Diagonalisation

```
A=np.matrix([[1,2,3],  
              [4,7,8],  
              [4,7,1]])  
  
w,P=np.linalg.eig(A)  
  
print(w)  
print(P)  
  
print(P*np.diag(w)*np.linalg.inv(P))
```

Application, diagonalisation

La matrice compagnon du polynome

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1} + x^n$$

$$\text{est } A = \begin{pmatrix} 0 & 0 & \cdots & 0 & -c_0 \\ 1 & 0 & \cdots & 0 & -c_1 \\ 0 & 1 & \cdots & 0 & -c_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_{n-1} \end{pmatrix}$$

Les valeurs propres de A sont les racines du polynome p

Application, diagonalisation

Construire la matrice compagnon des polynômes suivants, et en déduire leurs racines:

$$p_1(x) = x^2 - 1$$

$$p_2(x) = x^2 + 1$$

$$p_3(x) = x^3 + 2x^2 - 1$$

$$p_4(x) = x^6 - 11x^5 + 30x^4 - 12x^3 - 13x^2 - x - 42$$

Application, diagonalisation

Construire la matrice compagnon des polynomes suivants,
et en déduire leurs racines:

$$p_1(x) = x^2 - 1$$

$$p_2(x) = x^2 + 1$$

$$p_3(x) = x^3 + 2$$

$$p_4(x) = x^6 - 1$$

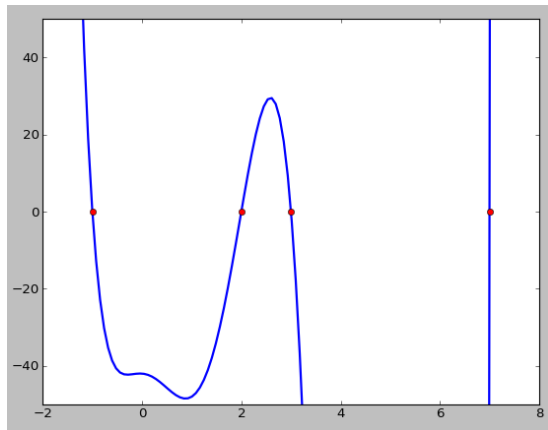
```
c=np.array([-42,-1,-13,-12,+30,-11])  
  
N=len(c)  
A=np.zeros([N,N])  
for k in range(N-1):  
    A[k+1,k]=1  
  
A[:,-1]=-c  
  
w,P=np.linalg.eig(A)
```

Application, diagonalisation

Afficher p_3 , ainsi que l'ensemble de ses racines réelles

Application, diagonalisation

Afficher p3, ainsi que l'ensemble de ces racines réelles



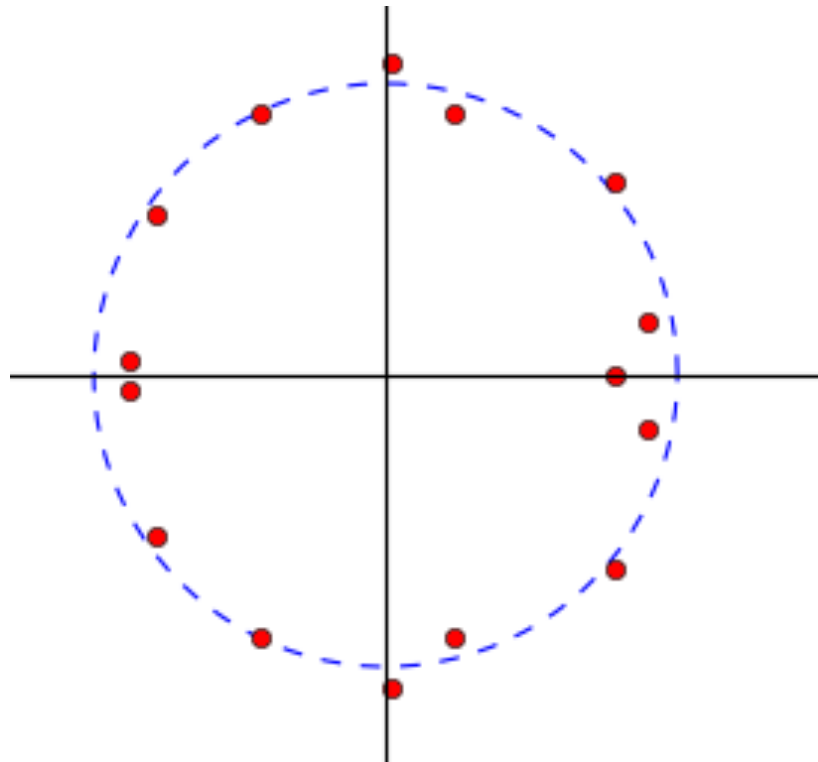
```
def f(x,c):  
    p=np.zeros(len(x))  
    for k in range(len(x)):  
        for i in range(len(c)):  
            p[k]+=c[i]*x[k]**i  
        p[k]+=x[k]**(len(c))  
    return p
```

```
x=np.linspace(-8,8,200)  
plt.plot(x,f(x,c),linewidth=2)  
epsilon=1e-5  
for r in w:  
    if(abs(r.imag)<epsilon):  
        plt.plot(r.real,0,"ro")  
plt.axis([-2,8,-50,50])  
plt.show()
```

Application, diagonalisation

Construire un polynome dont les coefficients sont des reels aléatoires

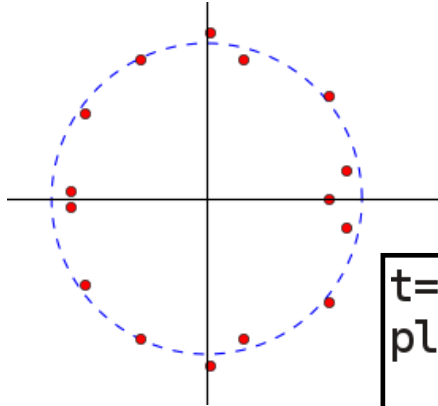
Observez la distribution des racines dans le plan complexe



Application, diagonalisation

Construire un polynome dont les coefficients sont des reels aléatoires

Observez la distribution des racines dans le plan complexe



```
t=np.linspace(0,2*np.pi,100)
plt.plot(np.cos(t),np.sin(t),"b--")

for r in w:
    plt.plot(r.real,r.imag,"ro")

plt.plot([-2,2],[0,0],"k")
plt.plot([0,0],[-2,2],"k")
plt.axis("equal")
plt.axis([-2,2,-2,2])
plt.show()
```

Affichage matrices/images

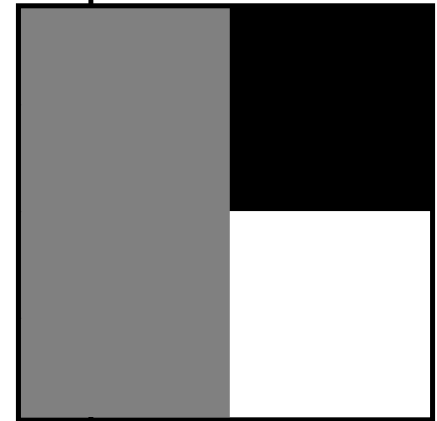
Affichage matrices/images

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

A=np.matrix([[1,0],
             [1,2]])

im=plt.imshow(A)

im.set_cmap('gray')
im.set_interpolation('nearest')
plt.show()
```



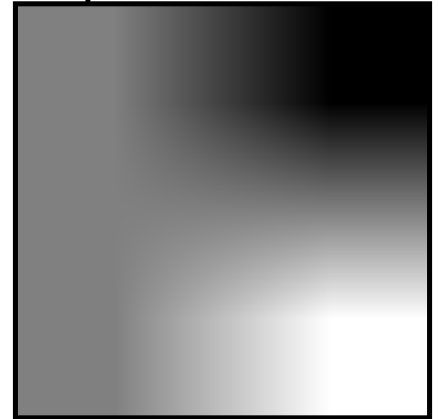
Affichage matrices/images

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

A=np.matrix([[1,0],
             [1,2]])

im=plt.imshow(A)

im.set_cmap('gray')
im.set_interpolation('nearest')
plt.show()
```



Affichage fonctions 2D

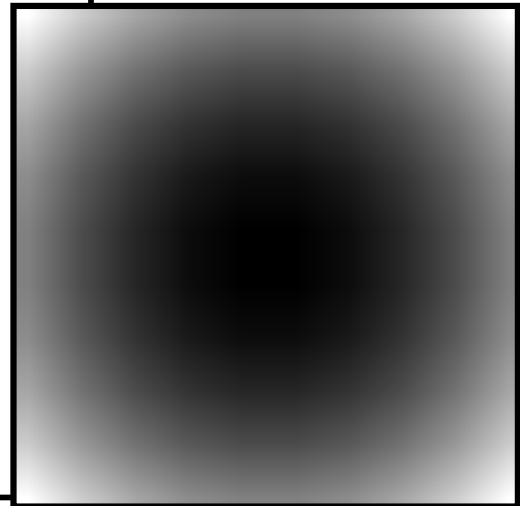
```
import numpy as np
import matplotlib.pyplot as plt
```

```
N=10
vx=np.linspace(-1.0,1.0,N)
vy=vx
```

```
z=np.zeros([N,N])
for kx,x in enumerate(vx):
    for ky,y in enumerate(vy):
        z[kx,ky]=x**2+y**2
```

```
plt.imshow(z)
plt.set_cmap("gray")
plt.show()
```

$$f(x, y) = x^2 + y^2$$



Affichage fonctions 2D

Soit:

$$\begin{cases} f(x, y) = \cos(h(x, y - 2) h(x, y + 2)) \\ h(x, y) = 2 \sqrt{x^2 + y^2} \end{cases}$$

Afficher f (utiliser la colormap "hot")

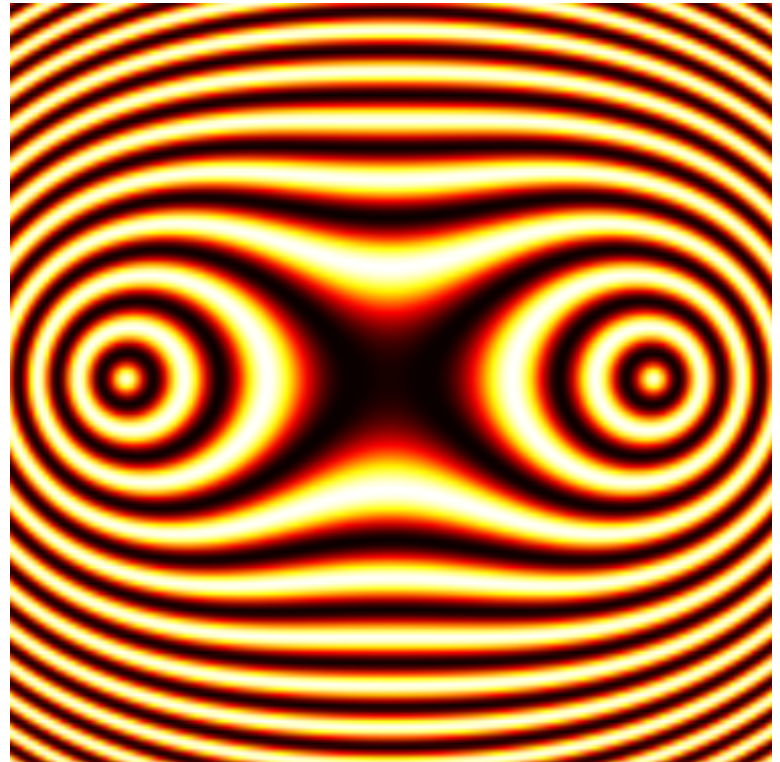
Affichage fonctions 2D

Soit:

$$\begin{cases} f(x, y) = \cos(h(x, y - 2) h(x, y + 2)) \\ h(x, y) = 2 \sqrt{x^2 + y^2} \end{cases}$$

Afficher f (utiliser la colormap "hot")

```
def h(x,y):  
    return 2*(x**2+y**2)**(0.5)  
  
def f(x,y):  
    z=np.cos(h(x,y-2)*h(x,y+2))  
    return z  
  
N=150  
vx,vy=np.linspace(-3,3,N),np.linspace(-3,3,N)  
  
z=np.zeros([N,N])  
  
for kx,x in enumerate(vx):  
    for ky,y in enumerate(vy):  
        z[kx,ky]=f(x,y)  
  
print(z)  
im=plt.imshow(z)  
im.set_cmap("hot")  
plt.show()
```



Application: Fractale

La fractale de Mandelbrot est obtenue en itérant la formule suivante:

$$\begin{cases} z_{k+1} = z_k^2 + c \\ z_0 = c \end{cases}$$

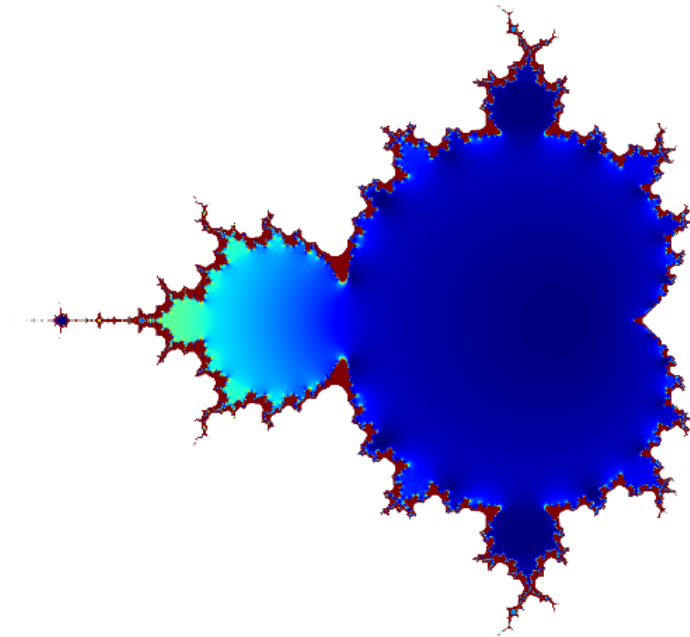
Afficher $|z|$ après N iterations pour $c \in [-2 - j, 2 + j]$

Application: Fractale

La fractale de Mandelbrot est obtenue en itérant la formule suivante:

$$\begin{cases} z_{k+1} = z_k^2 + c \\ z_0 = c \end{cases}$$

Afficher $|z|$ après N iterations pour $c \in [-2 - j, 2 + j]$



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

N=1500
u=np.linspace(-2,2,N)

xx,yy=np.meshgrid(u,u)
zz=xx+1j*yy
c=zz

for k in range(20):
    zz=zz**2+c

Z=zz.real**2+zz.imag**2

im=plt.imshow(Z)
im.set_clim(0,4)
plt.show()
```

Champs de vecteurs

Champs de vecteurs : quiver

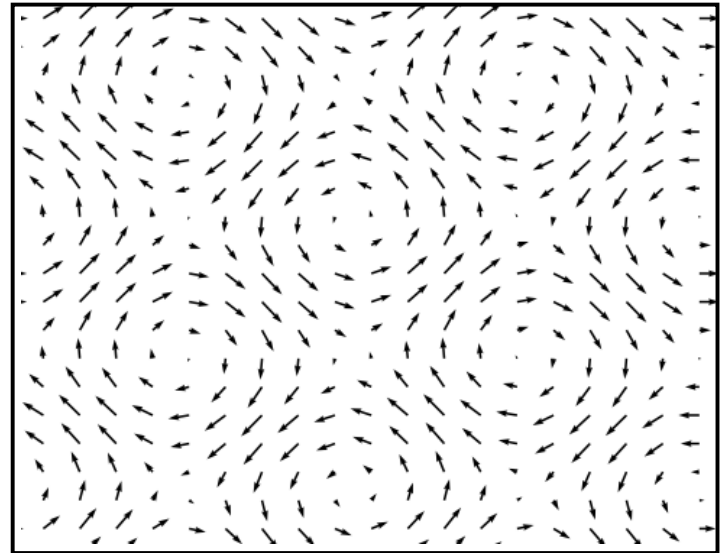
$$f : (x, y) \mapsto (\cos(x), \sin(y))$$

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

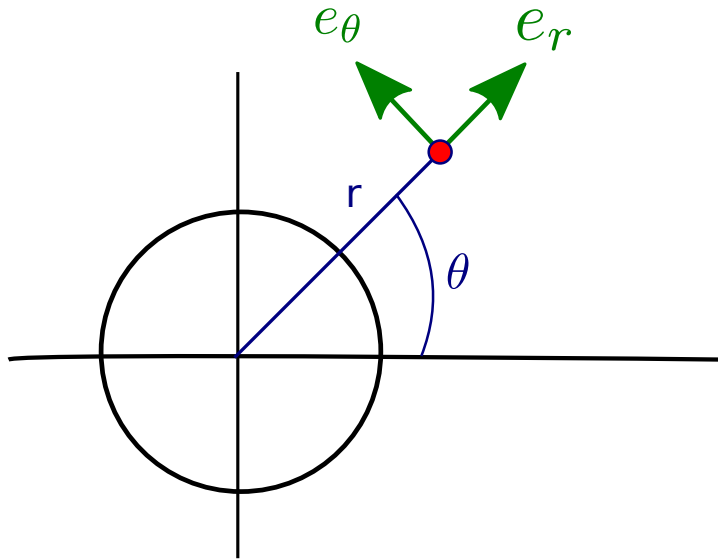
N=20
u=np.linspace(-1,1,N)

Vx=np.zeros([N,N])
Vy=np.zeros([N,N])
for kx,x in enumerate(u):
    for ky,y in enumerate(u):
        Vx[kx,ky]=np.cos(2*np.pi*x)
        Vy[kx,ky]=np.sin(2*np.pi*y)

plt.quiver(Vx,Vy)
plt.show()
```



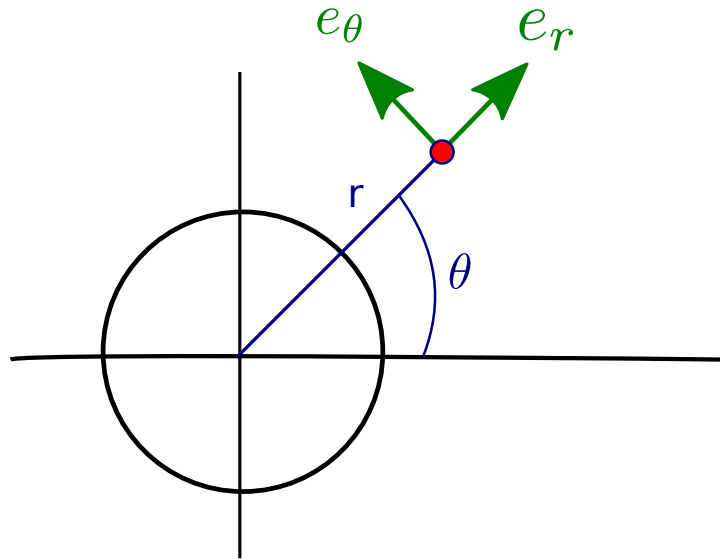
Changement de coordonnées



$$\begin{cases} e_x = \cos(\theta) e_r - \sin(\theta) e_\theta \\ e_y = \sin(\theta) e_r + \cos(\theta) e_\theta \end{cases}$$

Afficher e_θ et e_r pour $(x, y) \in [-1, 1]^2$

Changement de coordonnées

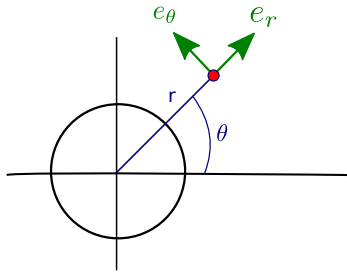


$$\begin{cases} e_x = \cos(\theta) e_r - \sin(\theta) e_\theta \\ e_y = \sin(\theta) e_r + \cos(\theta) e_\theta \end{cases}$$

Afficher

$$\begin{cases} f_r : (r, \theta) \mapsto (1, 0) \\ f_\theta : (r, \theta) \mapsto (0, 1) \end{cases}$$

Changement de coordonnées



$$\begin{cases} e_x = \cos(\theta) e_r - \sin(\theta) e_\theta \\ e_y = \sin(\theta) e_r + \cos(\theta) e_\theta \end{cases}$$

Afficher $\begin{cases} f_r : (r, \theta) \mapsto (1, 0) \\ f_\theta : (r, \theta) \mapsto (0, 1) \end{cases}$

```
xx,yy=np.meshgrid(u,u)
Vx=np.zeros([N,N])
Vy=np.zeros([N,N])

for kx,x in enumerate(u):
    for ky,y in enumerate(u):

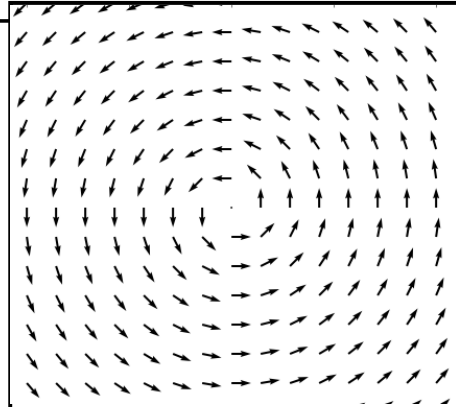
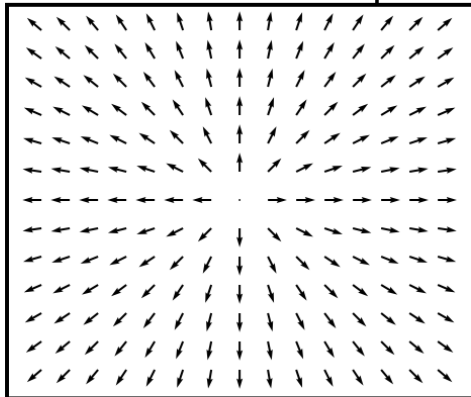
        r=(x**2+y**2)**0.5
        theta=math.atan2(y,x)

        if(r>=0.1):
            ur=1
            ut=0

            Vx[ky,kx]=np.cos(theta)*ur-np.sin(theta)*ut
            Vy[ky,kx]=np.sin(theta)*ur+np.cos(theta)*ut

im=plt.quiver(xx,yy,Vx,Vy)

plt.show()
```

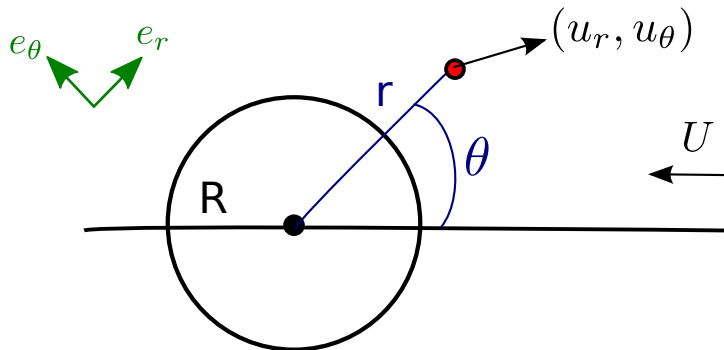


Application: mécanique des fluides

La vitesse d'un fluide (incompressible, non visqueux) autour d'une sphere de rayon R est donnée par

$$\begin{cases} u_r(r, \theta) = -U \cos(\theta) \left(1 - \frac{3R}{2r} + \frac{R^3}{2r^3} \right) \\ u_\theta(r, \theta) = U \sin(\theta) \left(1 - \frac{3R}{4r} - \frac{R^3}{4r^3} \right) \end{cases}$$

Afficher le champ de vecteur correspondant



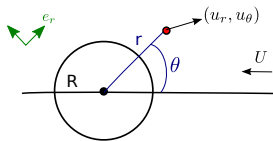
Application: mécanique des fluides

La vitesse d'un fluide (imcompressible, non visqueux) autour d'une sphère de rayon R est donné par

$$u_r(r, \theta) = -U \cos(\theta) \left(1 - \frac{3R}{2r} + \frac{R^3}{2r^3} \right)$$

$$u_\theta(r, \theta) = U \sin(\theta) \left(1 - \frac{3R}{4r} - \frac{R^3}{4r^3} \right)$$

Afficher le champ de vecteur correspondant



```
xx,yy=np.meshgrid(ux,uy)
Vx=np.zeros([N,N])
Vy=np.zeros([N,N])

for kx,x in enumerate(ux):
    for ky,y in enumerate(uy):

        r=(x**2+y**2)**0.5
        theta=math.atan2(y,x)

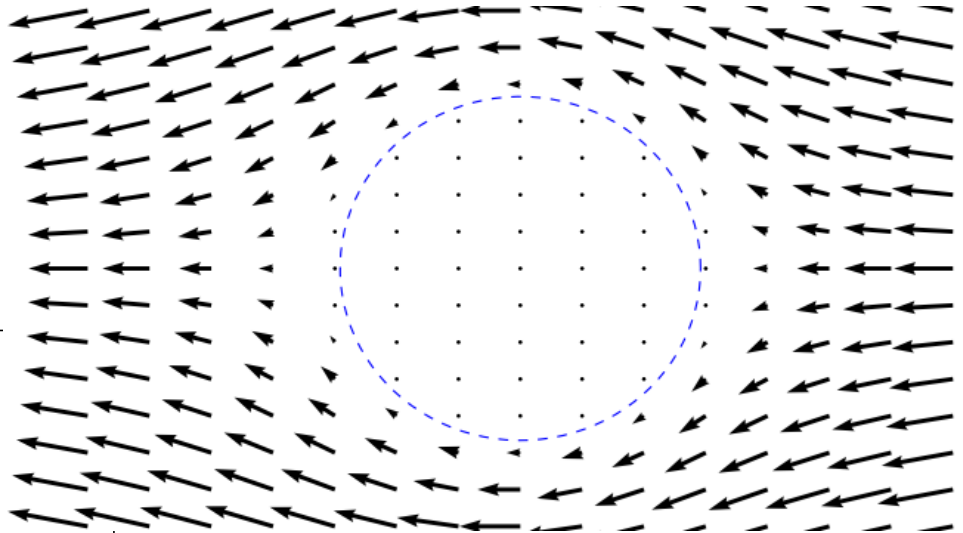
        if(r>=R):
            ur=-U*np.cos(theta)*(1-3*R/(2*r)+R**3/(2*r**3))
            ut= U*np.sin(theta)*(1-3*R/(4*r)-R**3/(4*r**3))

            n=np.linalg.norm([ur,ut])

            Vx[ky,kx]=-np.sin(theta)*ut+np.cos(theta)*ur
            Vy[ky,kx]= np.cos(theta)*ut+np.sin(theta)*ur

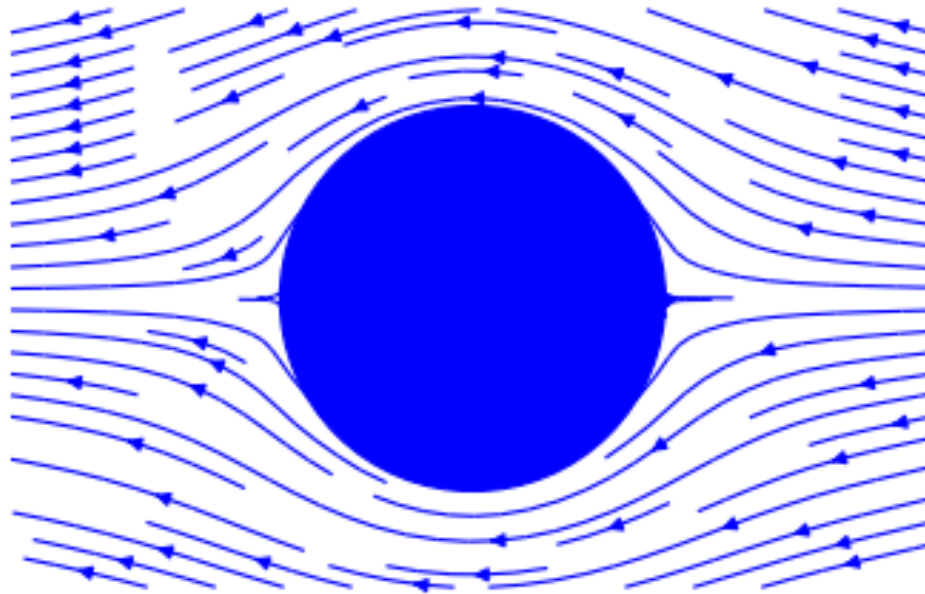
im=plt.quiver(xx,yy,Vx,Vy)

t=np.linspace(0,2*np.pi,100)
plt.plot(R*np.cos(t),R*np.sin(t),"b--")
plt.show()
```



Lignes de champs

```
plt.streamplot(xx, yy, Vx, Vy, density=1, color='b')
```



Entrées/sorties Fichiers

Chaine caracteres

Les chaines de caracteres sont des listes

```
mot="ma maison"
```

```
print(mot[0])
```

```
print(mot[2:6])
```

```
print(mot[::2])
```

```
for c in mot:  
    print(c)
```

m	a		m	a	i	s	o	n
0	1	2	3	4	5	6	7	8

Chaine caracteres

On ne modifie pas une chaine de caractere
(immutable)

```
mot="ma maison"
```

```
mot[2]="r"
```

'str' object does not support item assignment

Concatenation

+ concatène une chaîne avec une autre

```
mot_1="bonjour"  
mot_2="a toi"  
mot_3="!!"
```

```
mot_4=mot_1+mot_2+mot_3  
mot_5=mot_1+" "+mot_2+" "+mot_3  
mot_6=mot_1+" "+mot_2[2:]+mot_3
```

```
print(mot_4)
```

→ bonjoura toi!!

```
print(mot_5)
```

→ bonjour a toi !!

```
print(mot_6)
```

→ bonjour toi!!

Majuscule, minuscule

```
mot="ma maison"  
  
mot2=mot.replace("maison","demeure")  
mot3=mot.upper()  
mot4=mot[0].upper()+mot[1:].lower()  
  
print(mot2)  
print(mot3)  
print(mot4)
```

Séparation

Split sépare en plusieurs entités une chaîne à l'aide d'un délimiteur

```
chaîne="125+784-52.2+845=?"  
resultat=chaîne.split("+")  
  
print(resultat[0])  
print(resultat[1])  
print(resultat[2])
```

125+784-52.2+845=?
↑
resultat
[0] 125
[1] 784-52.2
[2] 845=?

Utilisation typique: espace

```
phrase="Maitre Corbeau, sur un arbre perche"  
  
mots=phrase.split(" ")  
  
for k,mot in enumerate(mots):  
    print("mot",k," :",mot)
```

```
mot 0 : Maitre  
mot 1 : Corbeau,  
mot 2 : sur  
mot 3 : un  
mot 4 : arbre  
mot 5 : perche
```

Application: nom/prenom

```
etudiant="jean dumont"  
  
mots=etudiant.split(" ")  
  
prenom=mots[0]  
nom=mots[1]  
  
prenom=prenom[0].upper()+prenom[1:].lower()  
nom=nom[0].upper()+nom[1:].lower()  
  
print(nom, ", ", prenom)
```

Dumont , Jean

Liste de mots

```
liste_mots=["maison","chocolat","elephant"]  
  
print(liste_mots[1][2])  
  
for mot in liste_mots:  
    print(mot[-1])
```

Application

Construire et afficher une liste de nom/prénoms telle que la liste:

```
etudiant=[ "jean dumont"  
»         , "FABRICE rene"  
»         , "Antoine GONTRAND"  
»         , "romain SEVERIN"]
```

Soit transformée en:

- liste ordonnée dans l'ordre alphabétique du nom
- le nom et le prenom sont séparés
- la première lettre du nom et prénom sont en majuscule (le reste en minuscule)

Dumont	Jean
Gontrand	Antoine
Rene	Fabrice
Severin	Romain

Application

Construire et afficher une liste de nom/prénoms telle que la liste:

```
etudiant=["jean dumont"  
»         ,"FABRICE rene"  
»         ,"Antoine GONTRAND"  
»         ,"romain SEVERIN"]
```

Soit transformée en:

- liste ordonnée dans l'ordre alphabétique du nom
- le nom et le prénom sont séparés
- la première lettre du nom et prénom sont en majuscule (le reste en minuscule)

Dumont	Jean
Gontrand	Antoine
Rene	Fabrice
Severin	Romain

```
nouvelle_liste=[]  
for e in etudiant:  
    prenom,nom=e.split(" ")  
    nom=nom[0].upper()+nom[1:].lower()  
    prenom=prenom[0].upper()+prenom[1:].lower()  
    nouvelle_liste.append([nom,prenom])  
  
nouvelle_liste=sorted(nouvelle_liste)  
print(nouvelle_liste)
```

Conversion nombre/mot

```
a=7  
b=12.2
```

```
mot_a=str(a)  
mot_b=str(b)
```

```
mot_c=mot_a+mot_b
```

```
c=float(mot_c)
```

```
d=c+a  
print(d)
```

Conversion en texte

Concaténation de
texte

Conversion texte en nombre

Addition de nombre

Application:

Extraire les notes de ce texte

Afficher la moyenne correspondante

```
notes="chimie:14.5 physique:9.5 math:8.7"
```

Application:

Extraire les notes de ce ce texte
Afficher la moyenne correspondante

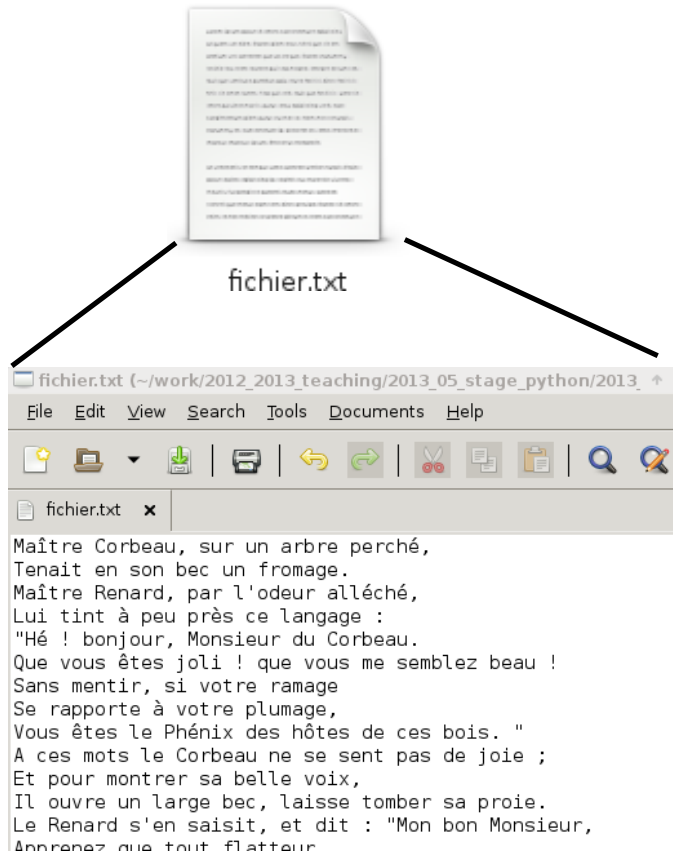
```
notes="chimie:14.5 physique:9.5 math:8.7"
```

```
avg=0.0
matiere=notes.split(" ")
for m in matiere:
    nom,note=m.split(":")
    avg+=float(note)

avg/=len(matiere)
print(avg)
```

Fichiers

Lecture d'un fichier



Lecture ligne à ligne

```
fichier=open("fichier.txt")  
  
for ligne in fichier:  
    print(ligne)  
  
fichier.close()
```

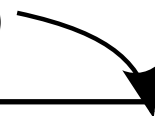
Traitement d'un texte

```
fichier=open("fichier.txt")  
  
for ligne in fichier:  
    nouvelle_phrase=ligne.replace("Corbeau","Pelican")  
    print(nouvelle_phrase)  
  
fichier.close()
```

Maître Pelican, sur un arbre perché,
Tenait en son bec un fromage.
Maître Renard, par l'odeur alléché,
Lui tint à peu près ce langage :
"Hé ! bonjour, Monsieur du Pelican."
...

Ecriture fichier

write/ecrit
(supprime/crée un fichier vierge)



```
fichier=open("mon_fichier.txt","w")  
fichier.write("Bonjour fichier \n")  
fichier.write("3+3="+str(3+3))  
  
fichier.close()
```

```
Bonjour fichier  
3+3=6
```

Application: Analyse données

Les fichiers disponibles fournissent les températures moyennes de Lyon en fevrier et avril 2011, 2012, et 2013.

Afficher et comparer les courbes de températures

Algorithme:

```
Pour chacun des fichiers
|   Pour chaque ligne à l'exception de la premiere
|   |   Separer les données suivant le symbol "/"
|   |   Lire la 4ème donnée en tant que nombre
|   |   Stocker cette valeur dans un vecteur de temperature
|   Afficher le vecteur de temperature
```

http://www.meteorologic.net/metar-climato_LFLY.html

Application: Analyse données

```
import matplotlib.pyplot as plt

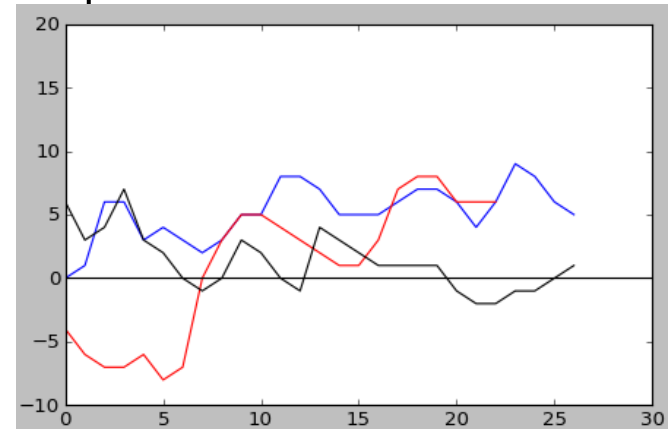
files=[open("../data/fevrier_2011_lyon.txt"),
        open("../data/fevrier_2012_lyon.txt"),
        open("../data/fevrier_2013_lyon.txt")]

temp=[]

for k_year,f in enumerate(files):
    temp.append([])
    for k_line,line in enumerate(f):
        if k_line>0:
            data=line.split("/")
            temp[k_year].append(float(data[3]))

plt.plot(temp[0],"b")
plt.plot(temp[1],"r")
plt.plot(temp[2],"k")
plt.axis([0,30,-10,20])

plt.show()
```



Application: Formatage

Soit le fichier suivant:

```
Nom;Prenom;Note Math;Note Physique;Note Chimie  
herz;jean;12;4.8;9.9  
hill;bejamin;9.5;12;12  
amenda;francis;15;14;11  
gorgie;andy;12;9.8;8
```

Ecrire le traitement qui viendra écrire le fichier suivant:

```
AMENDA francis 13.33  
GORGIE andy 9.93  
HERZ jean 8.9  
HILL bejamin 11.17
```

Application: Formatage

Soit le fichier suivant:

```
Nom;Prenom;Note Math;Note Physique;Note Chimie  
herz;jean;12;4.8;9.9  
hill;bejamin;9.5;12;12  
amenda;francis;15;14;11  
gorgie;andy;12;9.8;8
```

```
f=open("note_etudiant.txt")  
liste=[]  
fout=open("note_formatee.txt","w")  
for k_line,line in enumerate(f):  
    if k_line>0:  
        data=line.split(";")  
  
        nom=data[0]  
        prenom=data[1]  
  
        avg=0  
        for note in data[2:]:  
            avg+=float(note)  
        avg/=len(data[2:])  
        avg=round(avg,2)  
  
        liste.append([nom.upper(),prenom.lower(),avg])  
  
liste=sorted(liste)  
for etudiant in liste:  
  
    fout.write(etudiant[0]+" ")  
    fout.write(etudiant[1]+" ")  
    fout.write(str(etudiant[2])+"\n")
```

Programmes externes

Programmes externes

Le module "os" permet de communiquer avec le système d'exploitation

os.system(...) appelle un programme externe

```
import os  
os.system("firefox http://prepas.org/")
```