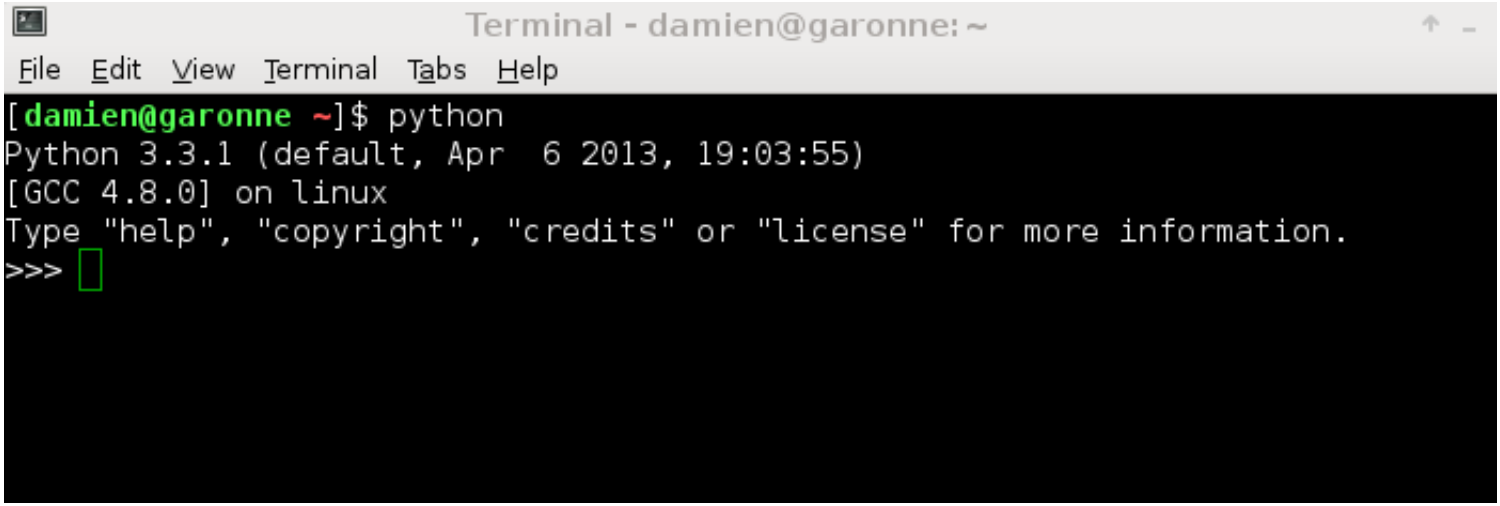
# Introduction à la programmation

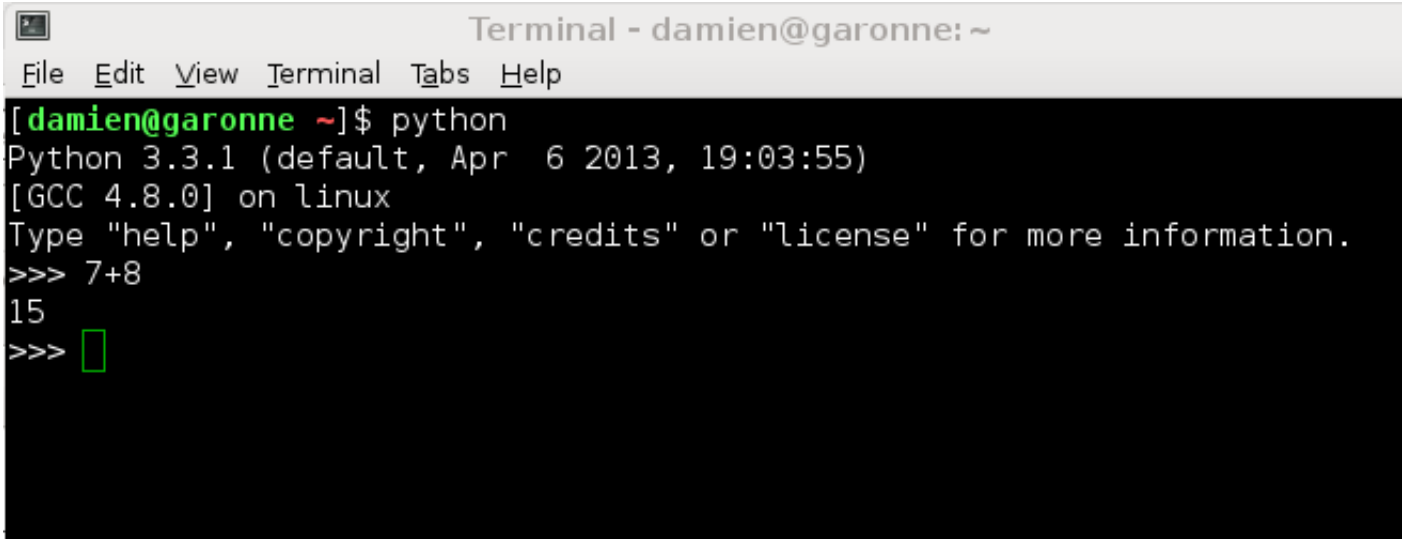
## Python


*Damien Rohmer*

# Premier "programme"



# Premier "programme"



# 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

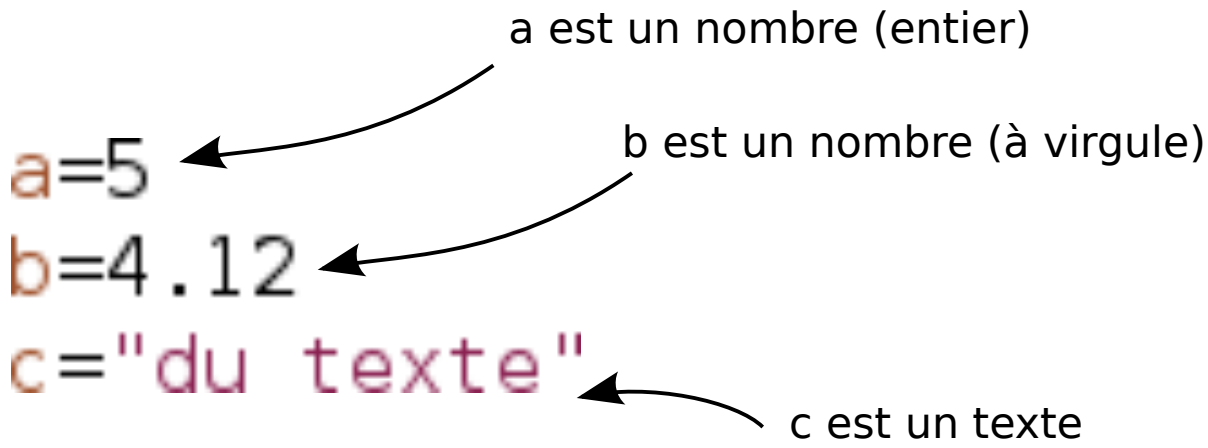
Affichage à l'écran:

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

> le resultat de 2+2 vaut 4

# Commandes

Types de variables:




a+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:

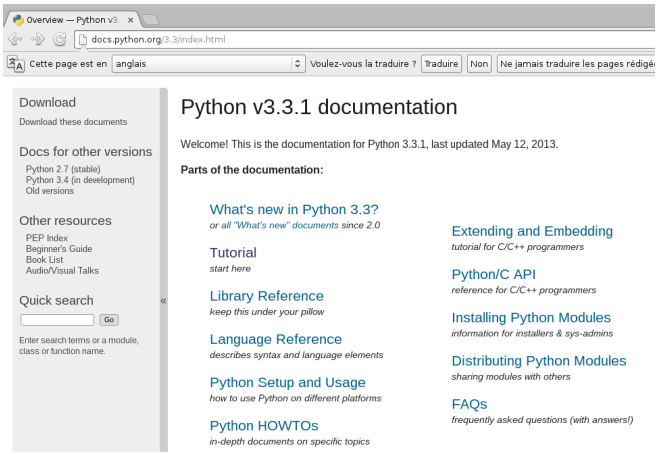
**help(***nom_fonction***)**

*ex.* help(pow)

Site web:

*http://docs.python.org/2/index.html*

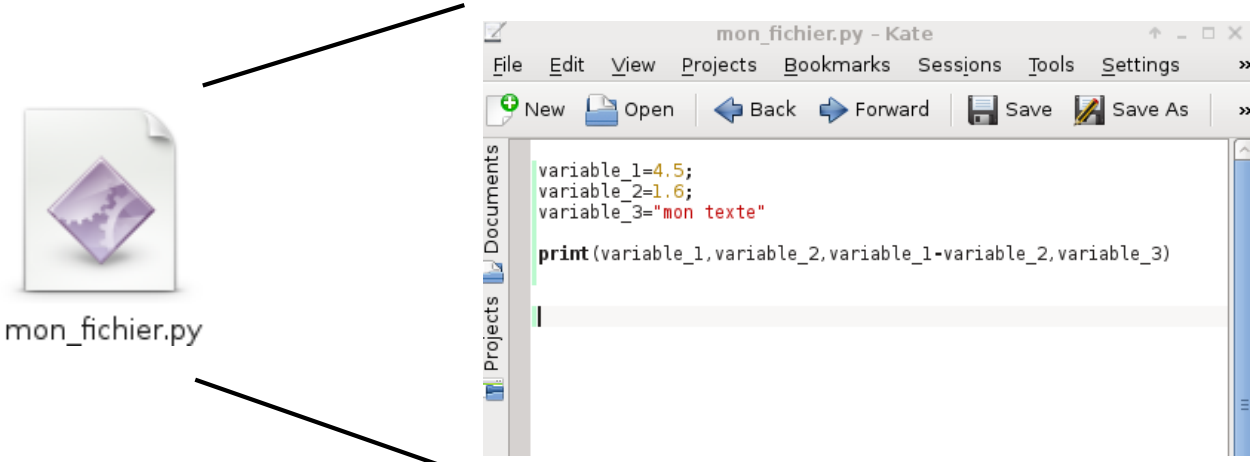
*http://docs.python.org/3/index.html*



# Ecriture dans un fichier

Ecrire ligne à ligne est fastidieux ...

On écrit d'abord dans un fichier texte




*(.py = fichier texte lisible par Python)*

On lance Python sur le fichier

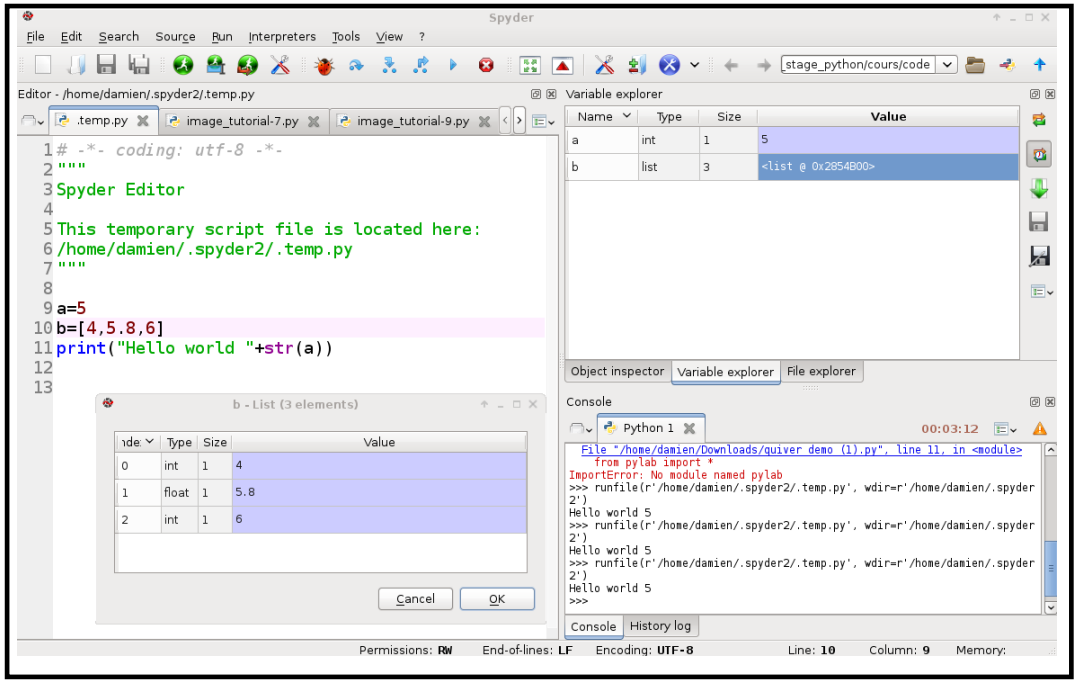
```
[damien@damien_pc ~/work/2012_2013_teaching
ython/cours/code]$ python mon_fichier.py
4.5 1.6 2.9 mon texte
```

# Editeur Python

Editeur de texte (attention à l'indentation)

- Linux: Kate
- Window: par defaut, pyscripter

Editeur type Matlab: **Spyder**



# Python: le langage

Création en 1990 (C ~ 1973)

Scripts, manipulation texte, pas de scientifique

Module Numpy en 2005

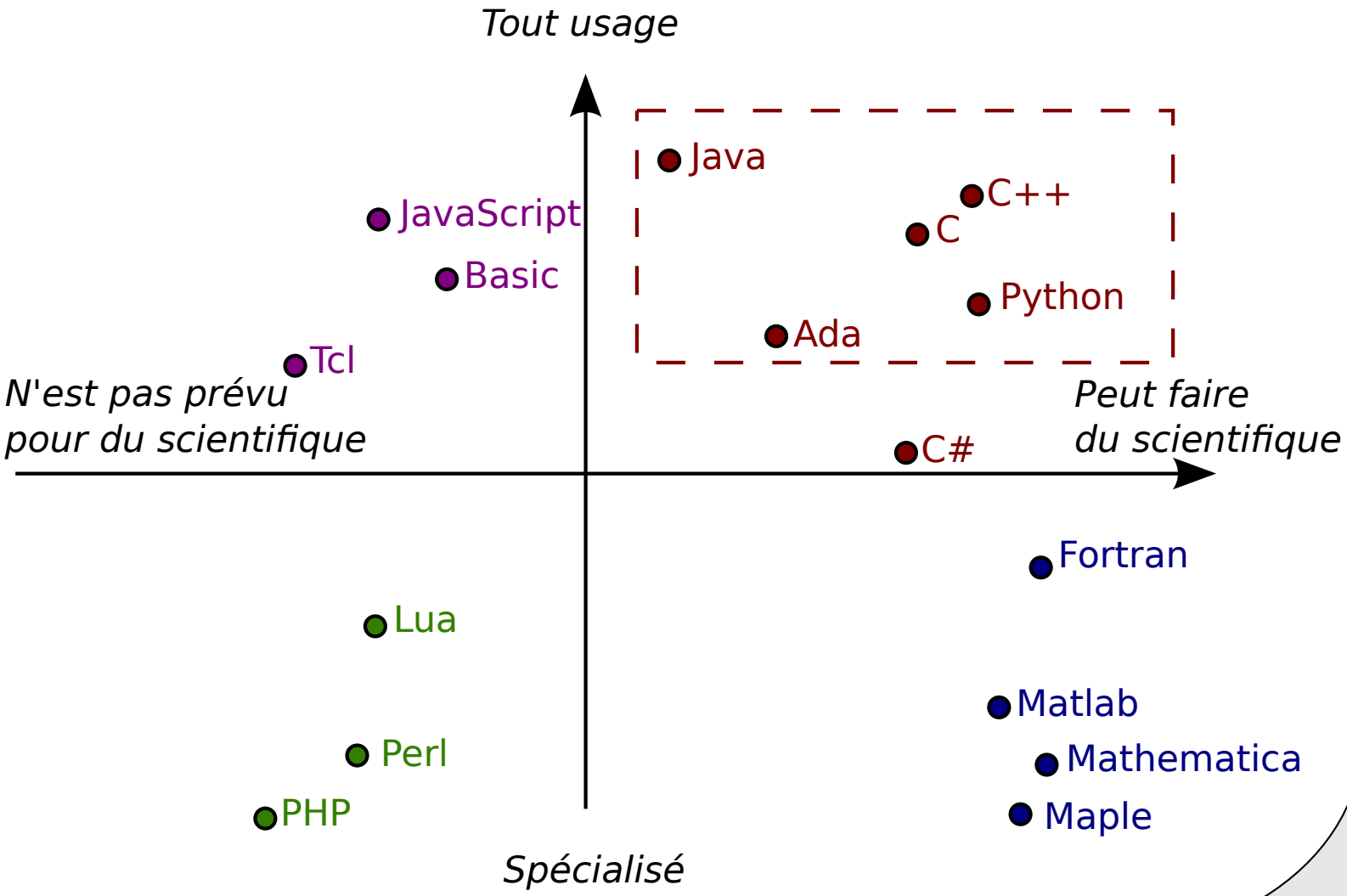
Developpement du calcul scientifique

Python 2.0 en 2000
Python 3.0 en 2009

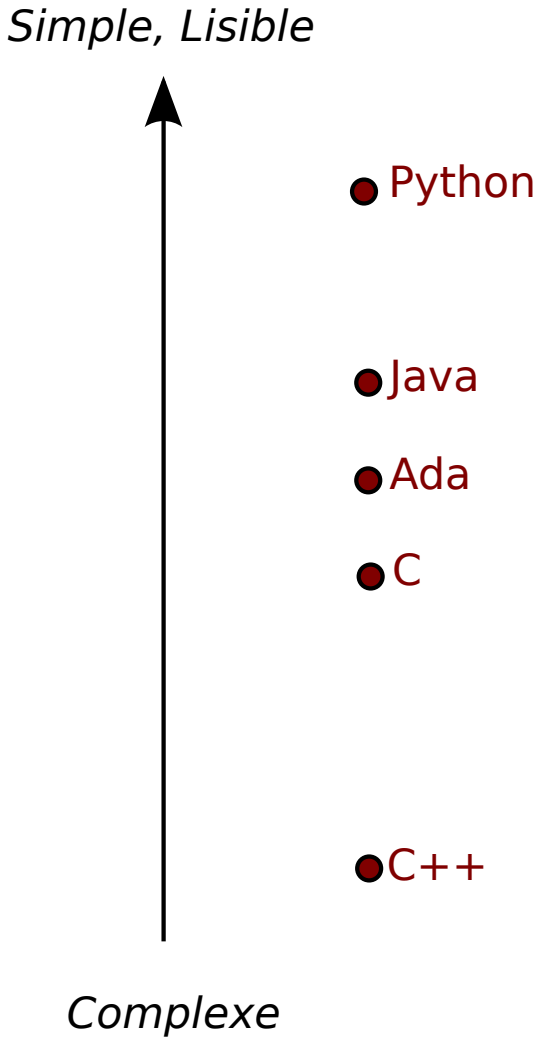

Python devient un acteur majeur du monde du calcul scientifique

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

# Python: positionnement

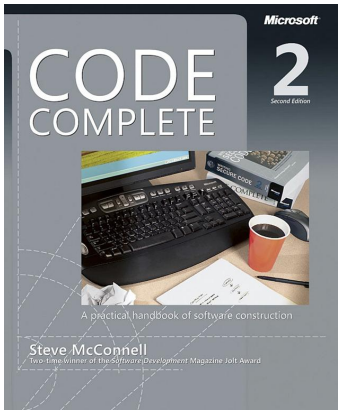


# Python: positionnement



# Python: Les + / -

**+ Langage très lisible**




*Lisibilité d'un code = Le + Important*

Un code est beaucoup plus lue qu'écrit
Le code est sa propre documentation
Erreur facilement détectable = gain de temps

# + Langage très lisible

Java

```java
public class MonTest{
    public static void main(string[] args){
        for(int i=0;i<10000;i++)
            System.out.println(i);
    }
}
```

Python

```python
for x in range(10000):
    print(i)
```

*Ce qui est simple*
*s'écrit simplement*

# Python: Les + / -

+ Langage très lisible

+ Algorithme proche du langage *(apprentissage)*

+ Utilisé en industrie — *script*, *calcul*

- Pas d'apprentissage "hardware"/OS

- Délicat pour code très volumineux

*typage dynamique*

# Python VS

| | |
|---|---|
| C | + Aisance codage, clareté<br>- Pas de contrôle bas niveau (embarqué, OS)<br>- Lent |
| C++ | + Clareté<br>- Lent |
| Java | + Simple, moins verbeux<br>+ Applicable science (operateurs)<br>- Moins répendu |
| Matlab | + Vraie informatique, structures données<br>+ Rapide<br>- Moins "sucre syntaxique" |

# Conditions: si, sinon

# Condition *if*

"si"

```
a=5
b=5

if a*b > 22:
  print("j'affiche ce message")
```

*(si a fois b est plus grand que 22)*

```
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

if a*b > 22:
    print("j'affiche ce message")
```

*: début d'un bloc de traitement*

*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 | |
|---|---|---|
| $a := b$ | a=b | affectation |
| $a = b$ | a==b | test d'égalité *(vaut vrai ou faux)* |

# 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  [4,5]           ")
else:
  print("intervalle ]5,+inf[")
```

# Condition *if* imbriqués

Quelle courbe dessine y si x varie ?

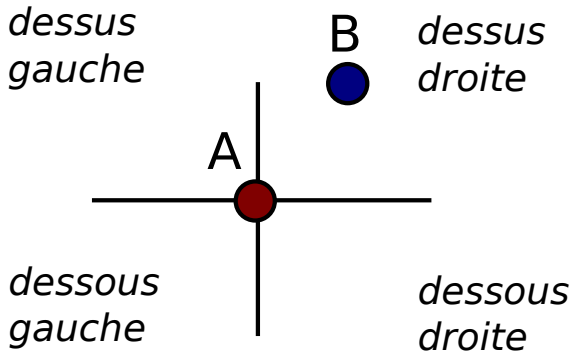
```
if x>=0:
   if x<1:
      y=x*x
   else:
      y=2x-1
else:
   if x>-1:
      y=-x*x
   else:
      y=2x+1
```

# Condition *if* imbriqués

## Application:

Soit 2 points A=(x1,y1) et B=(x2,y2)

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



# L'indentation

```
a,b=1,2
x,y=4,1
if a>5:
    x=x+2
y=y+3

print(y)
```

!=

```
a,b=1,2
x=4
if a>5:
    x=x+2
    y=y+3

print(y)
```

L'indentation fait partie du code!!!

Débat
philosophique
... religieux

```
a,b=1,2
x=4
if a>5:
    x=x+2
     y=y+3

print(x)
```

Ne s'execute pas!

IndentationError: unexpected indent

# L'indentation

```
a,b=1,2
x,y=4,1
if a>5:
    x=x+2
y=y+3

print(y)
```

!=

```
a,b=1,2
x=4
if a>5:
    x=x+2
    y=y+3

print(y)
```

\+ Force à la lisibilité du code

*Bon pour l'apprentissage*

+/- Copié-coller difficile

*Mais: copié-coller est à éviter*

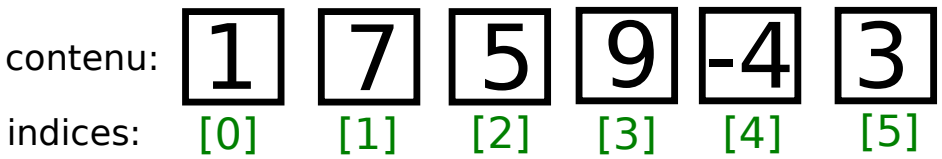
\- Portabilité: compatibilité Tab, espaces, ... (éditeur obligatoire)

# Les listes d'éléments

# Ensemble éléments: Listes

crochets [ ... ] indiquent une liste

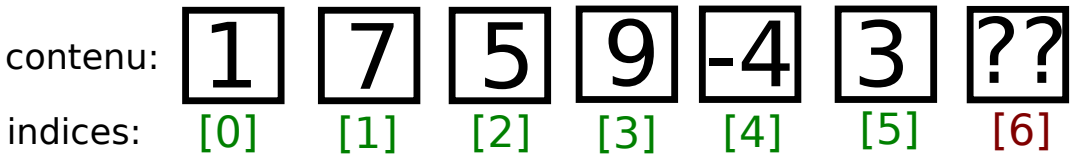
```
vecteur=[1,7,5,9,-4,3]

print(vecteur[0])
print(vecteur[1])
print(vecteur[2])

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

vecteur[2]=vecteur[4]*vecteur[5]
print(vecteur)
```



# Ensemble éléments: Listes

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

print(vecteur[6])
```

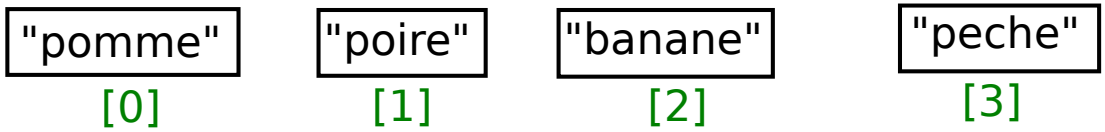
*IndexError: list index out of range*



# Ensemble éléments: Listes

Une liste peut contenir des mots

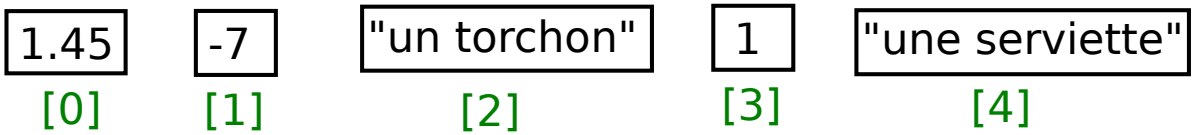
```
vecteur=["pomme","poire","banane","peche"]

vecteur[0]=vecteur[1]+" "+vecteur[2]
print(vecteur)
```



# 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)
```



# 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))
```

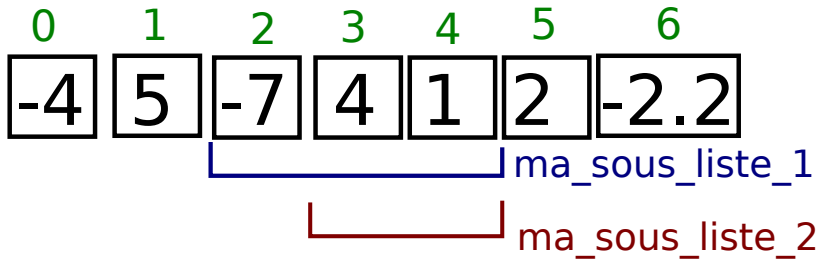
# Ensemble éléments: Listes

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)
```

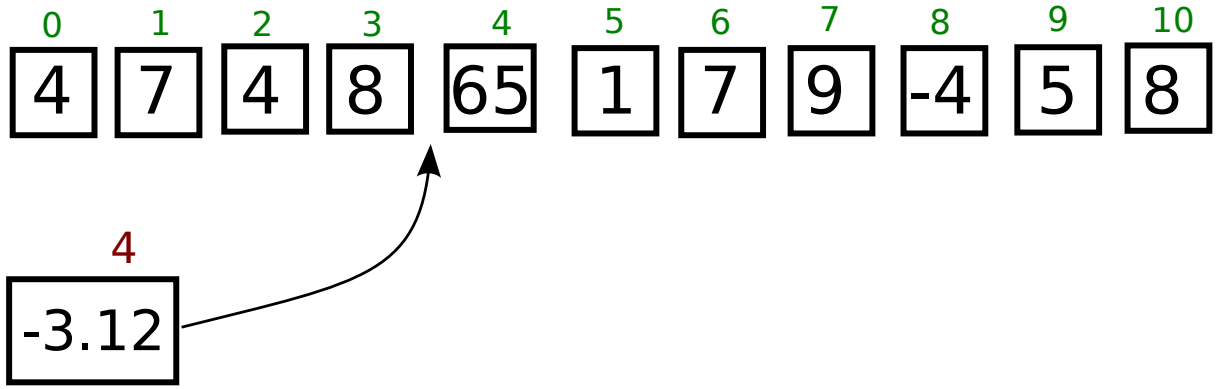
*unorderable types*
*str()<int()*

# 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 par valeur

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

Recherche valeur et supprime l'élément

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

# Ensemble éléments: Listes

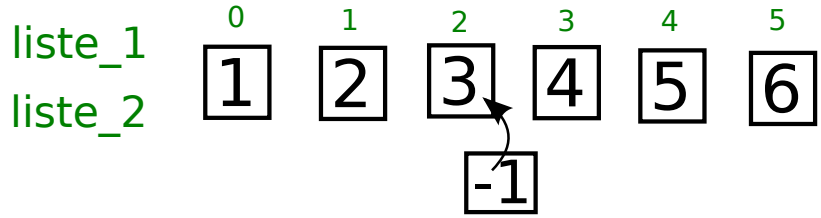
Exercice:

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)
```

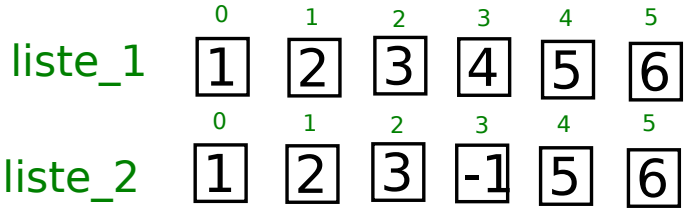


# Copie de liste

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)
```

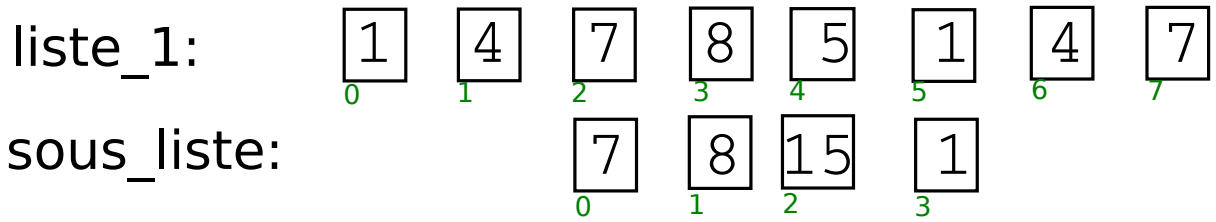
création
d'une copie de la liste



# Copie de liste

```
liste_1=[1,4,7,8,5,1,4,7]

sous_liste=liste_1[2:6]
sous_liste[2]=15

print(sous_liste)
print(liste_1)
```

Copie "automatique" par sous liste



Itération sur les listes

le mot clé "*for*"

# Créer des listes

$$L = \{f(k) | k \in [\![0, N[\![\}$$

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

En *francais*:

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

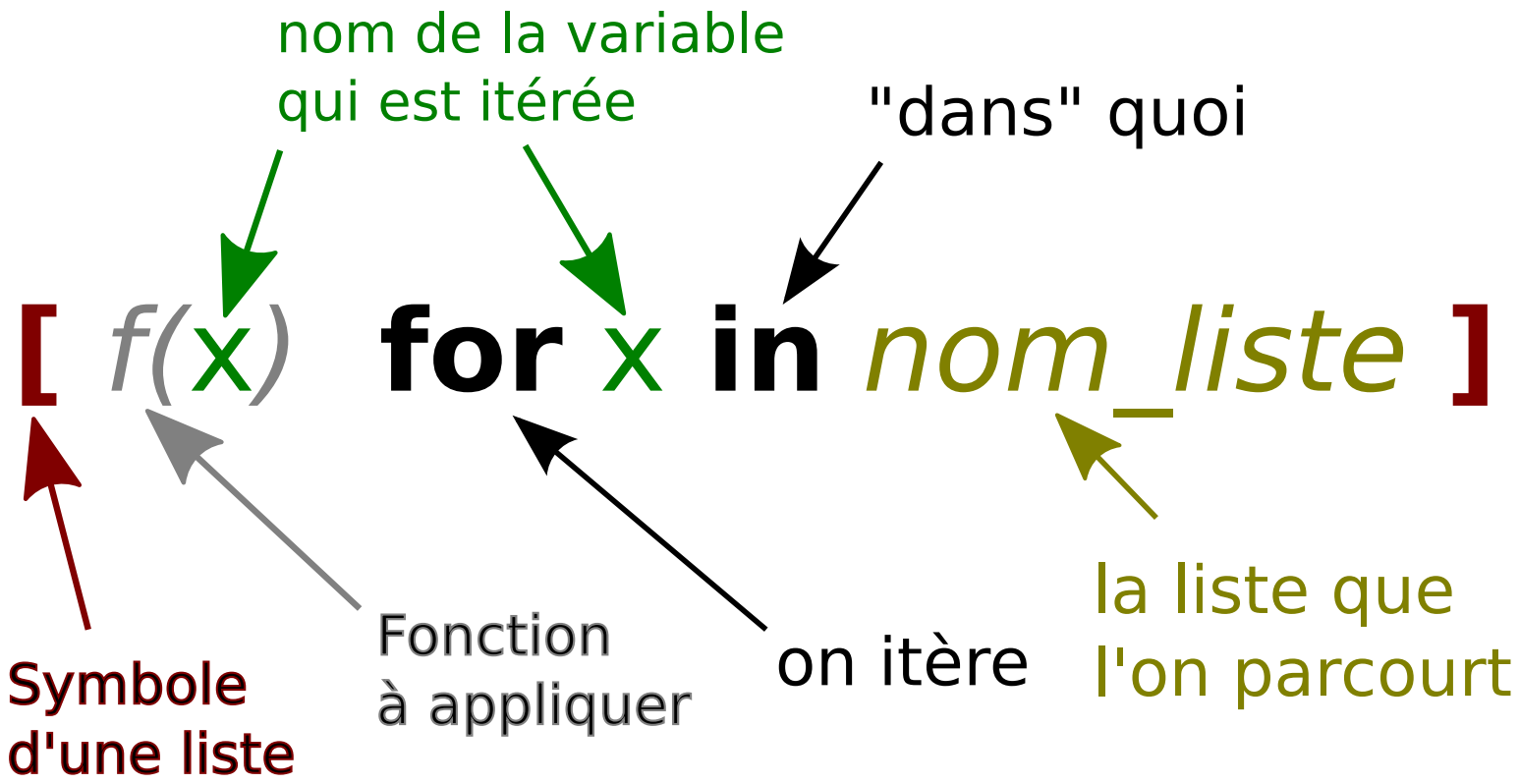
En *code*

```
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

Calculer:

$$L = \{a_k \mid k \in [\![-N, N[\![\}$$

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

*(|x| : abs(x))*

# Application

Calculer:

$$L = \{a_k \mid k \in [\![-N, N[\![\}$$

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

*(|x| : abs(x))*

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

# Application

$$L = \{a_k \mid k \in [\![-N, N[\![\}$$

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

```
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

$$L = \{a_k \mid k \in [\![-N, N[\![\}$$

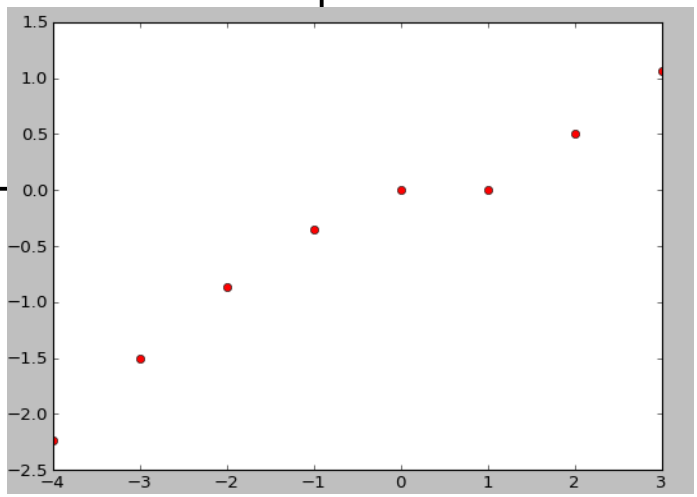
$$a_k = k\sqrt{|k-1|}/4$$

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:

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

$$f(k) = \dots$$

```
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)= ...

# Les fonctions

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

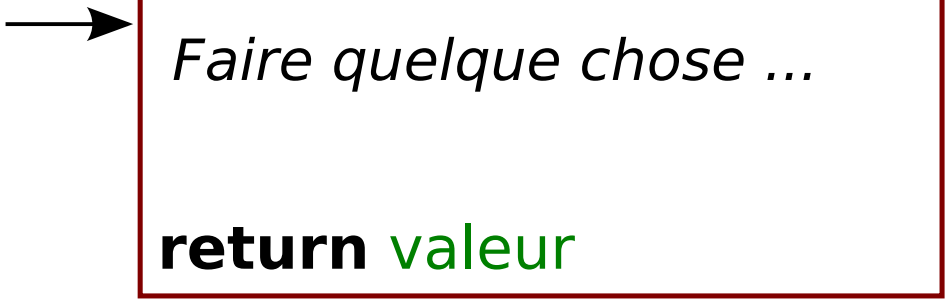
$$f(k) = k\sqrt{|k-1|}/4$$

```
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):



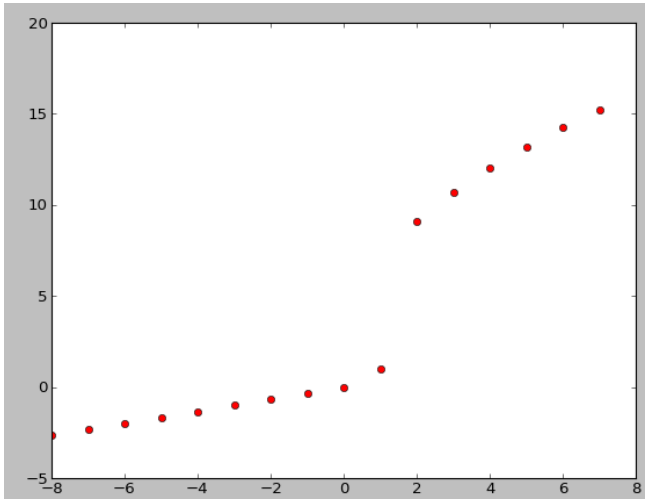

```
def f(k):
  return k*abs(k-1)**0.5/4


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

# Les fonctions

Les fonctions peuvent être *compliquées*

$$\begin{cases} f(k) = 5\sqrt{k} + 2 & k \geq 2 \\ f(k) = k^2 & k \in ]-1, 2[ \\ f(k) = k/3 & k \leq -1 \end{cases}$$

```
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 : retour d'arguments

```
def f(x):
    return [x[0]**2,x[1]**2,"chien"]

a = f([1,4])
print(a)

x0,x1,s = f([1,4])
print(x0,x1,s)

b,c = f([1,4])
print(b," , ",c)
```

*retour "liste"*

*récupère chaque argument (unpack)*

→ Erreur

*ValueError: too many values to unpack*

# Multi fichiers

ma_lib.py

```
import math

def norm(x):
    return math.sqrt(x[0]**2+x[1]**2)

def norm3(x):
    return (x[0]**3+x[1]**3)**(1/3.0)
```

```
import ma_lib

x=[1,8]
n2=ma_lib.norm(x)
n3=ma_lib.norm3(x)

print(n2,n3)
```

namespace automatique
(pas de collisions de noms)

# Multi fichiers

ma_lib.py

```
import math

def norm(x):
    return math.sqrt(x[0]**2+x[1]**2)

def norm3(x):
    return (x[0]**3+x[1]**3)**(1/3.0)
```

```
from ma_lib import norm3

x=[1,8]
#n2=norm(x)
n3=norm3(x)

print(n3)
```

*non accessible*

*plus de namespace*

# Multi fichiers

ma_lib.py

```
import math

def norm(x):
    return math.sqrt(x[0]**2+x[1]**2)

def norm3(x):
    return (x[0]**3+x[1]**3)**(1/3.0)
```

```
from ma_lib import *

x=[1,8]
n2=norm(x)
n3=norm3(x)

print(n3)
```

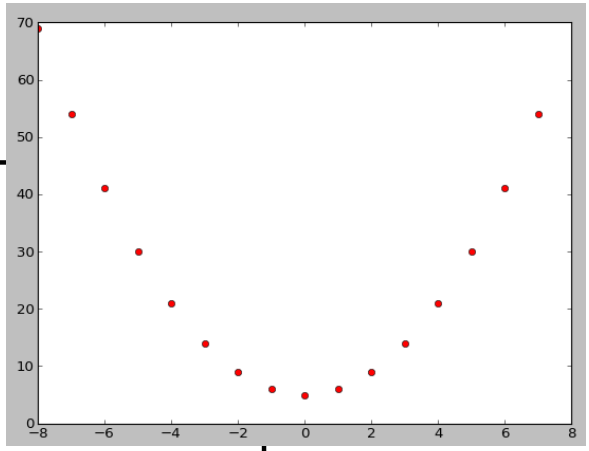
*Tout est accessible*

A éviter

# 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 etre vectorielles

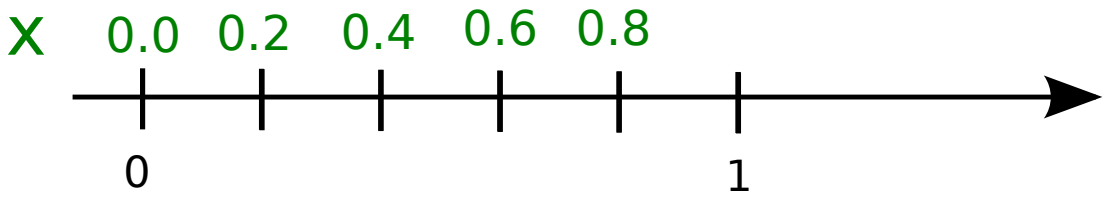
$$\begin{array}{cc} \mathbb{R}^3 & \mathbb{R}^2 \\ f : (x, y, z) & \mapsto \left(x^2 + y, xy\right) \end{array}$$

```
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)
```

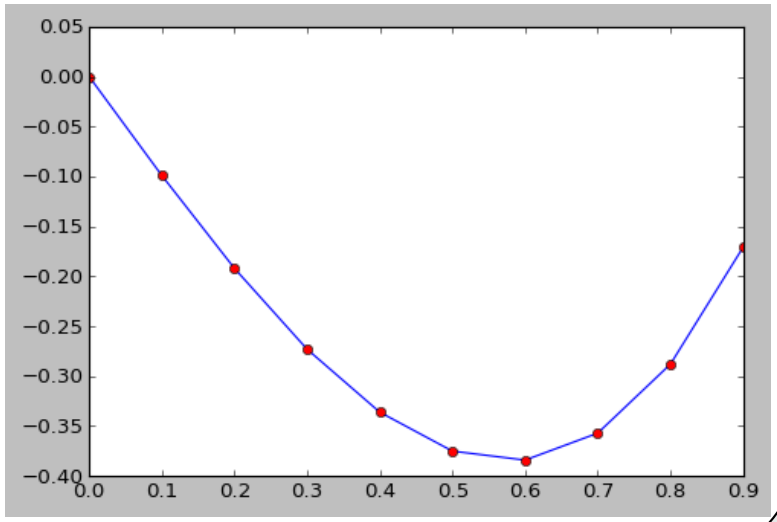
# Echantillonner dans $\mathbb{R}$

Calculer et stocker les N échantillons de f(x)

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

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


```
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 $\mathbb{R}$

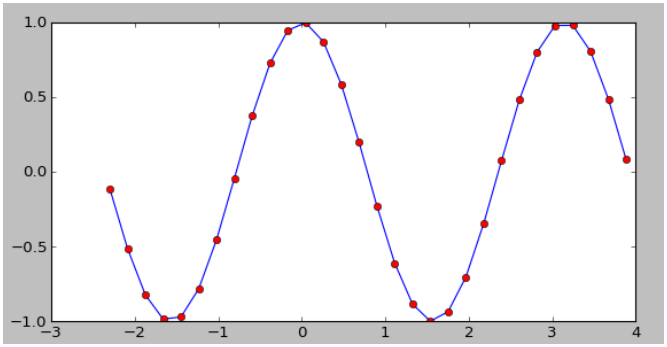
Calculer N échantillons de f pour $x \in [a, b[$

$$f(x) = \cos(2x)$$
$$[a, b[ = [-2.3, 4.1[$$

```
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()
```

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

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



# 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 nA
q=1.6 $10^{-19}$ C
k=1.38 $10^{-23}$ J/K

Tracer la caractéristique pour T=25° et T=65°
(298K) (338K)

$$V \in [-0.1, 0.4[V$$

Rem: Pour exponentielle

```
from math import exp

exp(x)
```

# 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 francais:

**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 francais:

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

En code:

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

True

False

*(type booléen)*

# 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

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

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

# Cas d'application

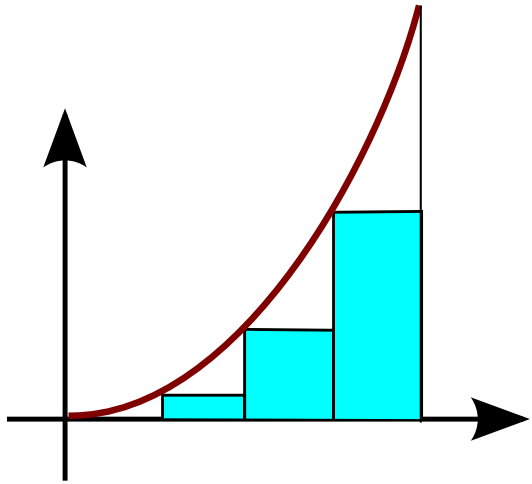
# Calcul intégral

# Calcul intégral

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

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

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



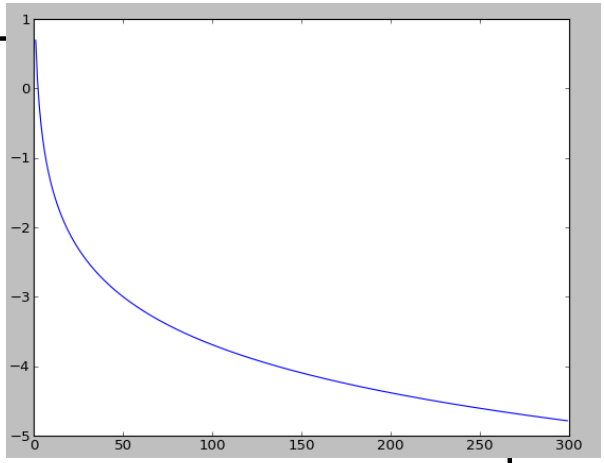
Calculer cette intégrale pour a=3, b=2, c=1 (prendre N=100)

# Calcul intégral

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

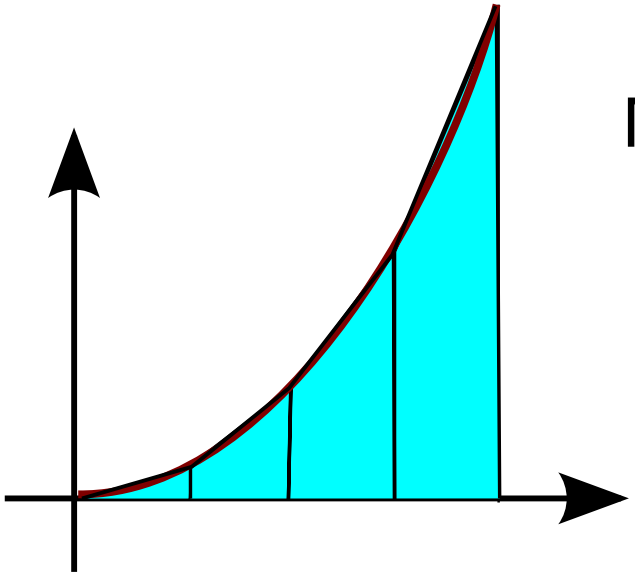
Tracer le log de l'erreur en fonction de N

```
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

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



Meilleure approximation

# Calcul intégral

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

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

# Boucle for "avancée"

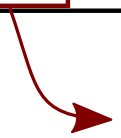
# Boucle sur des mots

```
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_matieres=["math","physique","chimie","informatique"]
ensemble_notes=[12.1,8.4,12.3,7.8]

[print(matiere,":",note) for matiere , note
 in zip(ensemble_matieres,ensemble_notes)]
```

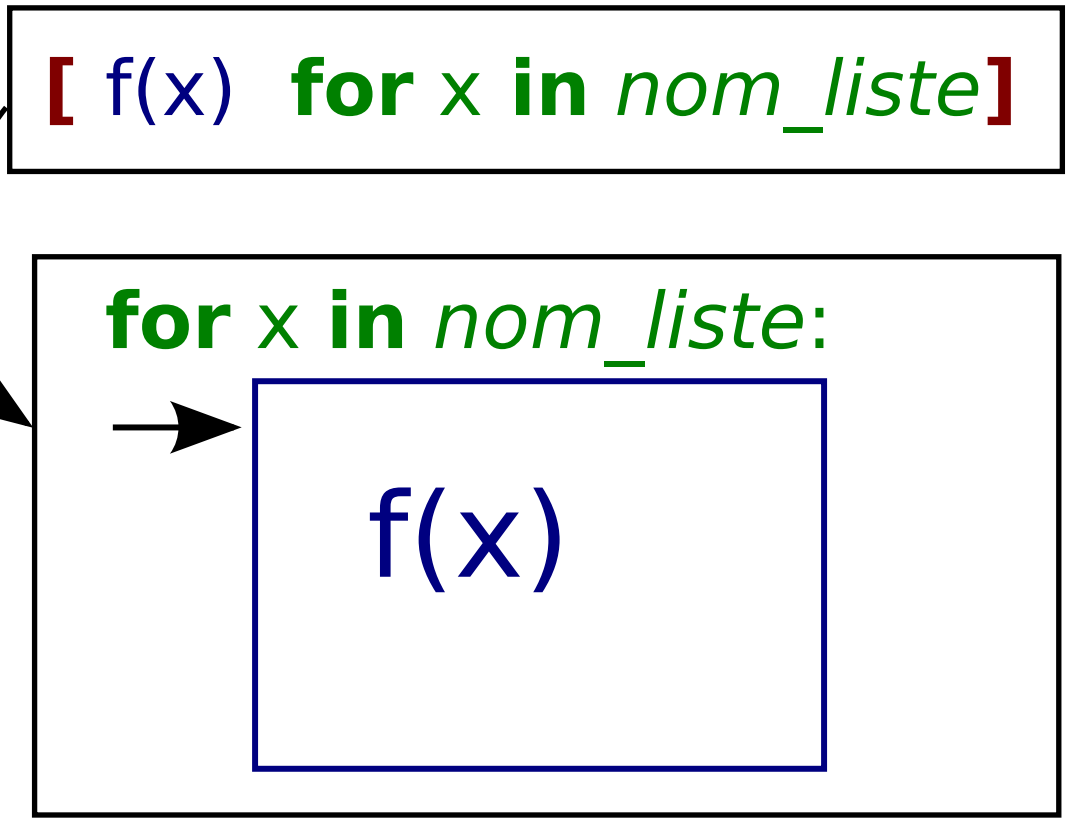


met les éléments ensemble

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



# 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_matieres=["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

# 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)
```

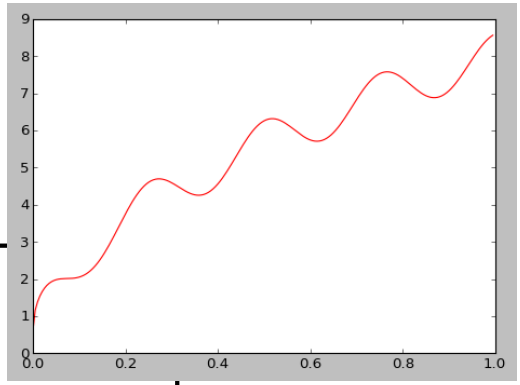
```
0 9.1
1 8.4
2 16.3
3 7.8
```

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

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

$$\forall k \in [\![0, N[\![ \,,\; a_k := f(k/N)$$

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

# 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 f(x)=E(x)%2 E: partie entière (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

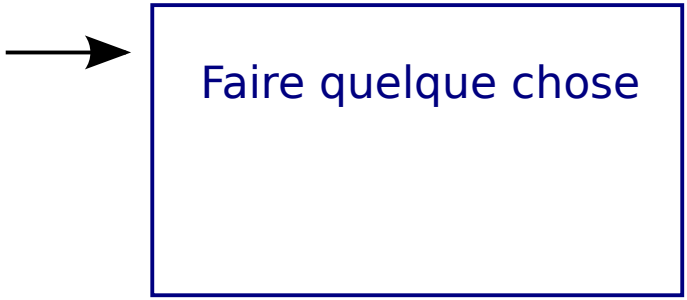
Itérer le processus sur z ... 15 fois

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

# Boucle while / "tant que"

# Boucle while / "tant que"

**while** *condition_vraie* **:**




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

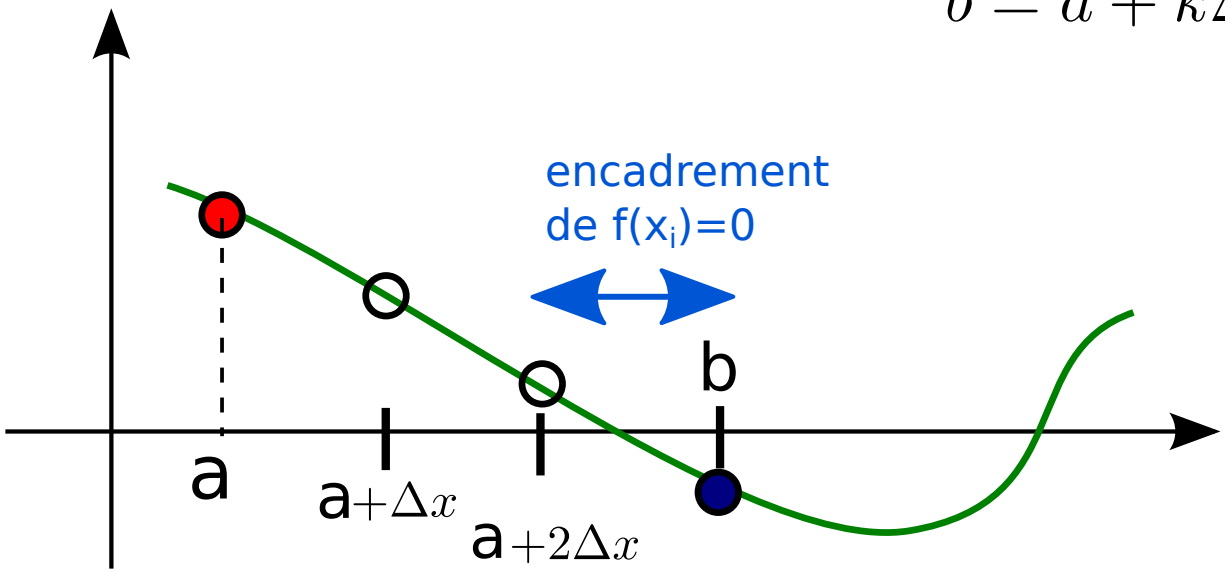
On ne connait pas forcément le nombre d'itérations

# Application: encadrement

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

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

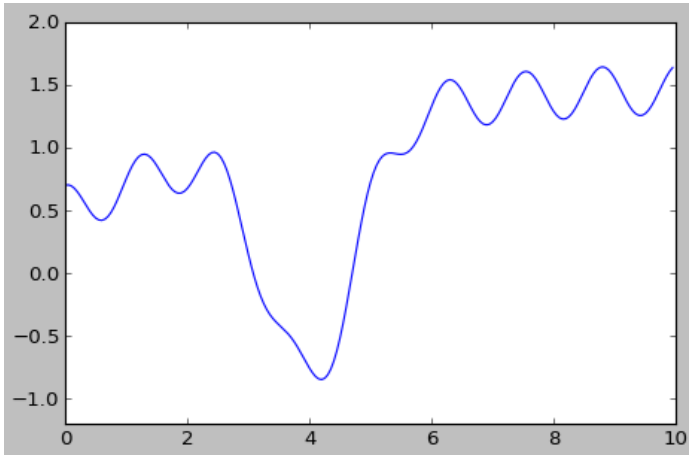
$$b = a + k\Delta x$$

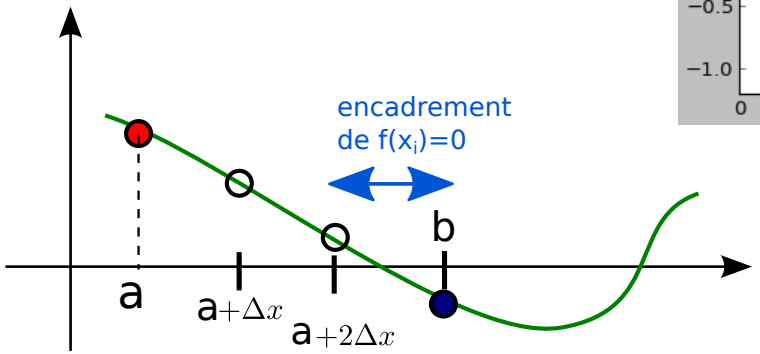


# Application: encadrement

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

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



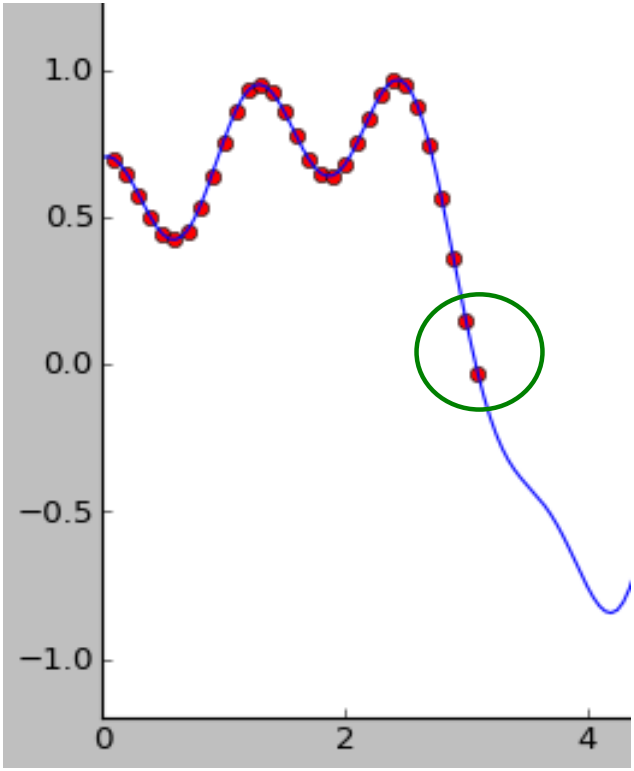



# Application: encadrement

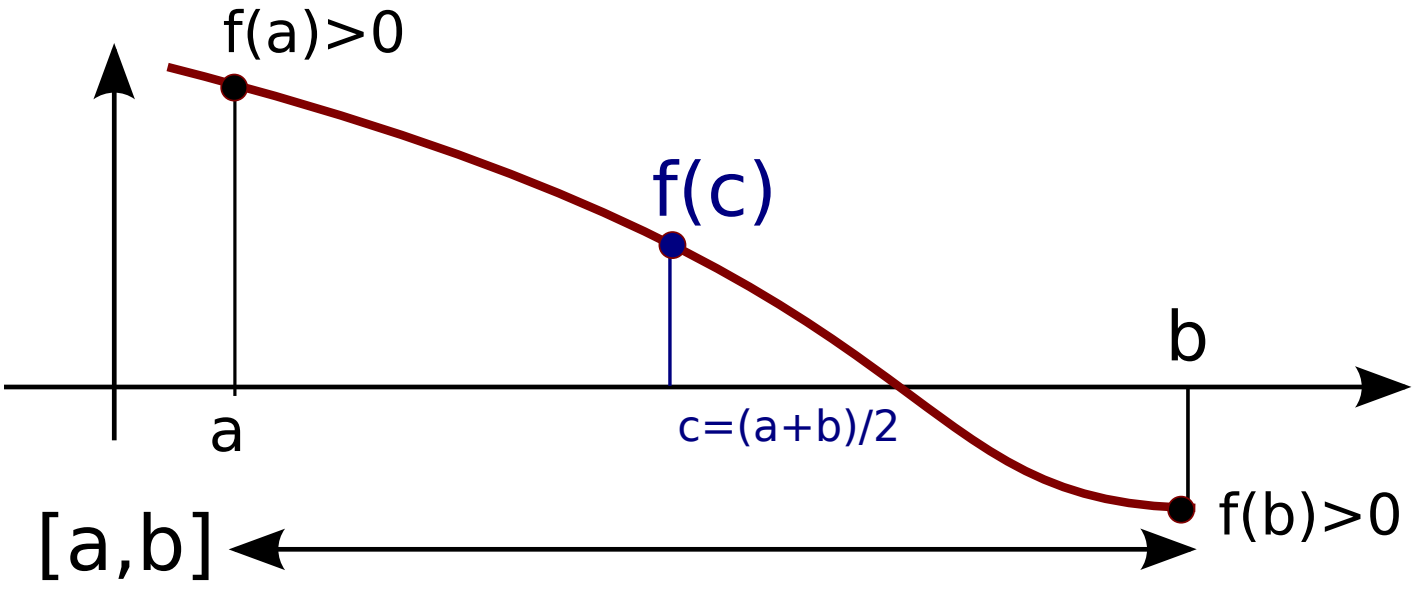
$$f(x) = \frac{1}{2} + \tanh\left(\frac{x}{5}\right) - 2e^{(-(x-4)^2)} + 0.2\cos(5x)$$

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

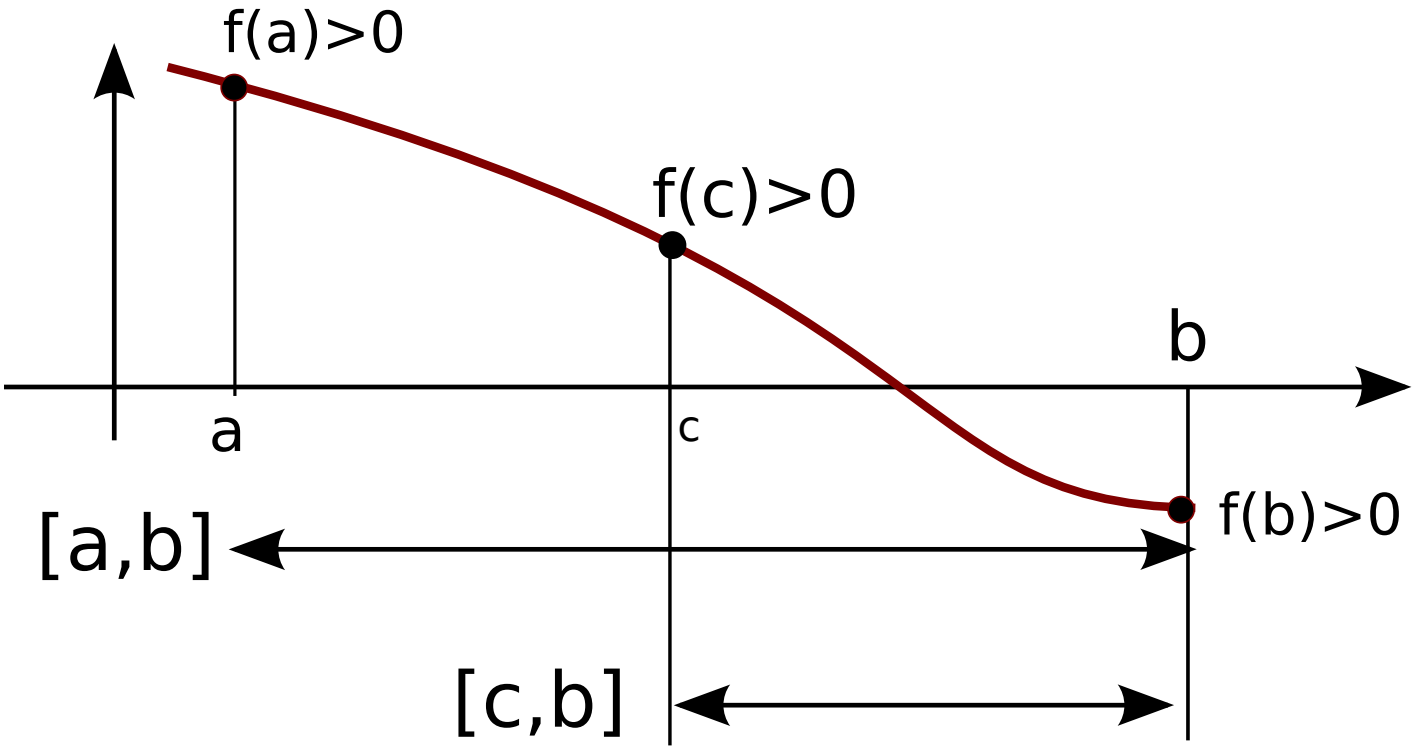
```
dx=0.1
a=0

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



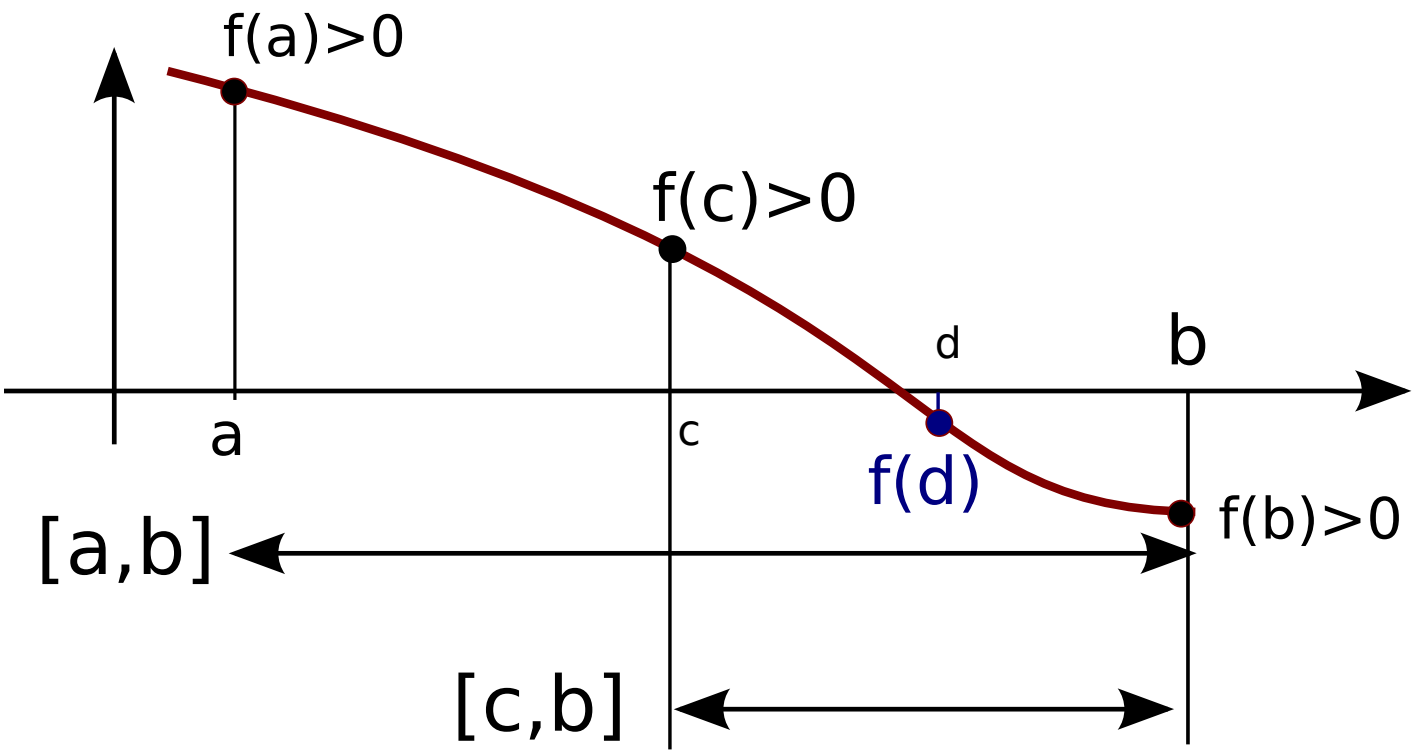
# Dichotomie



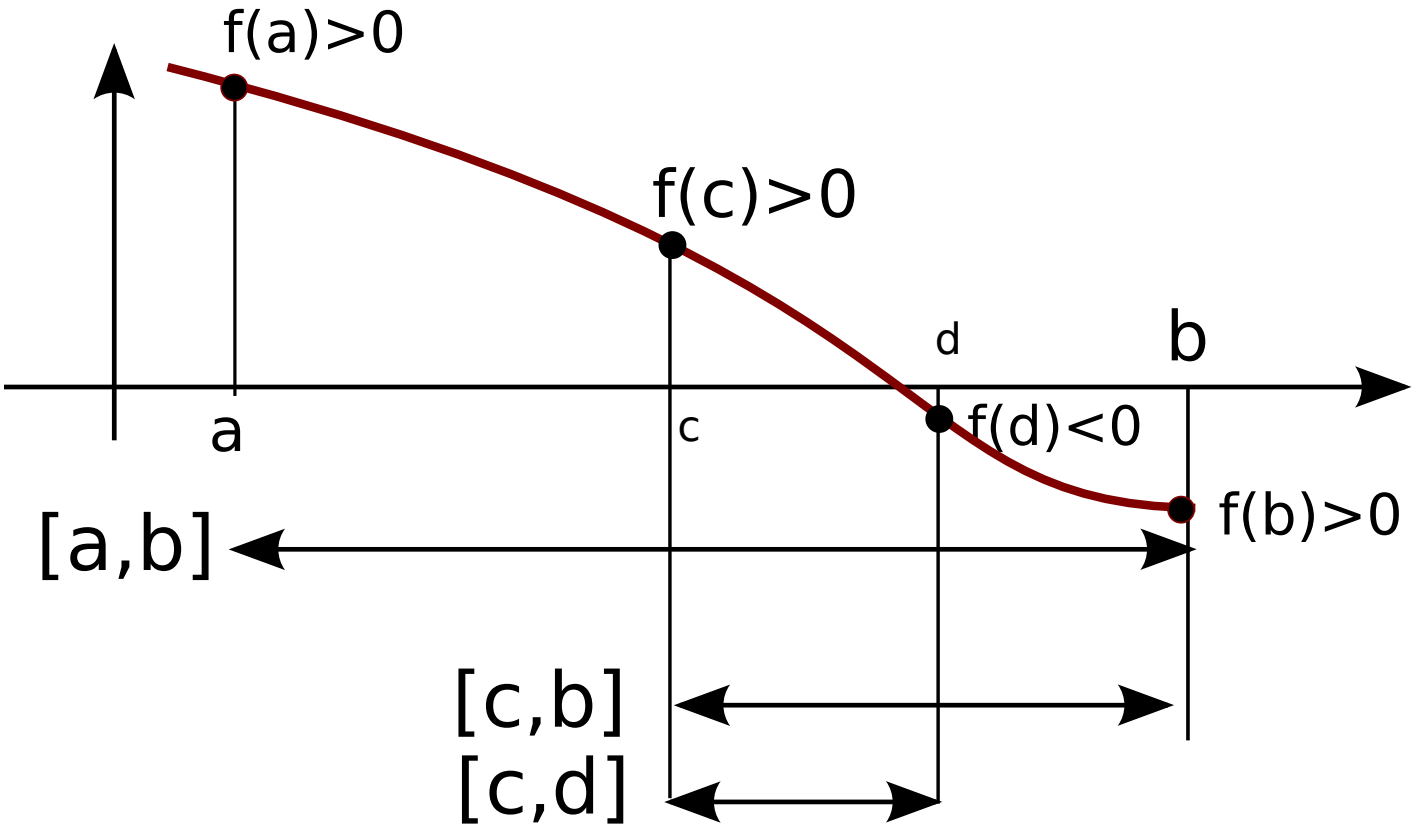
# Dichotomie



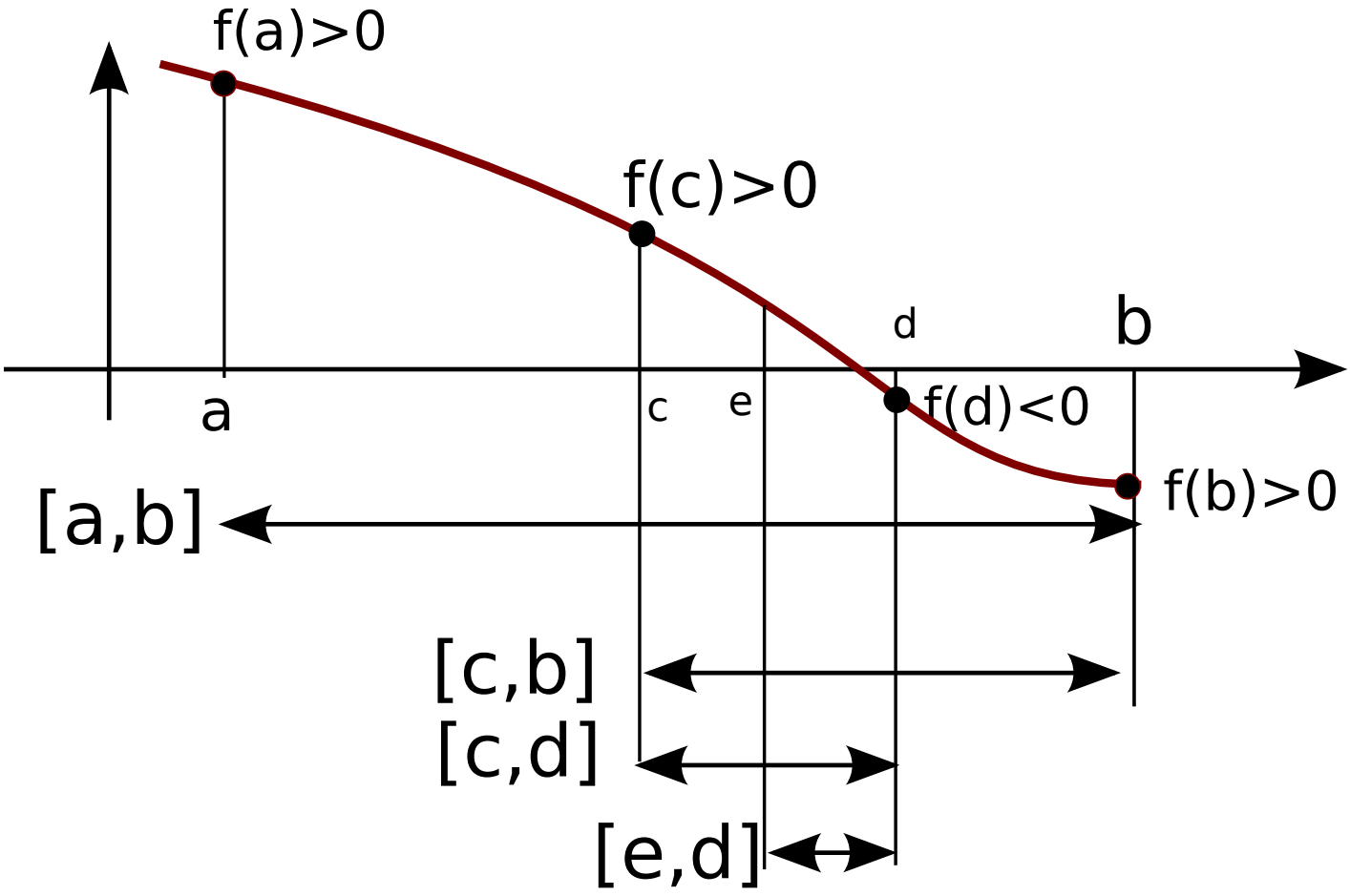
# Dichotomie



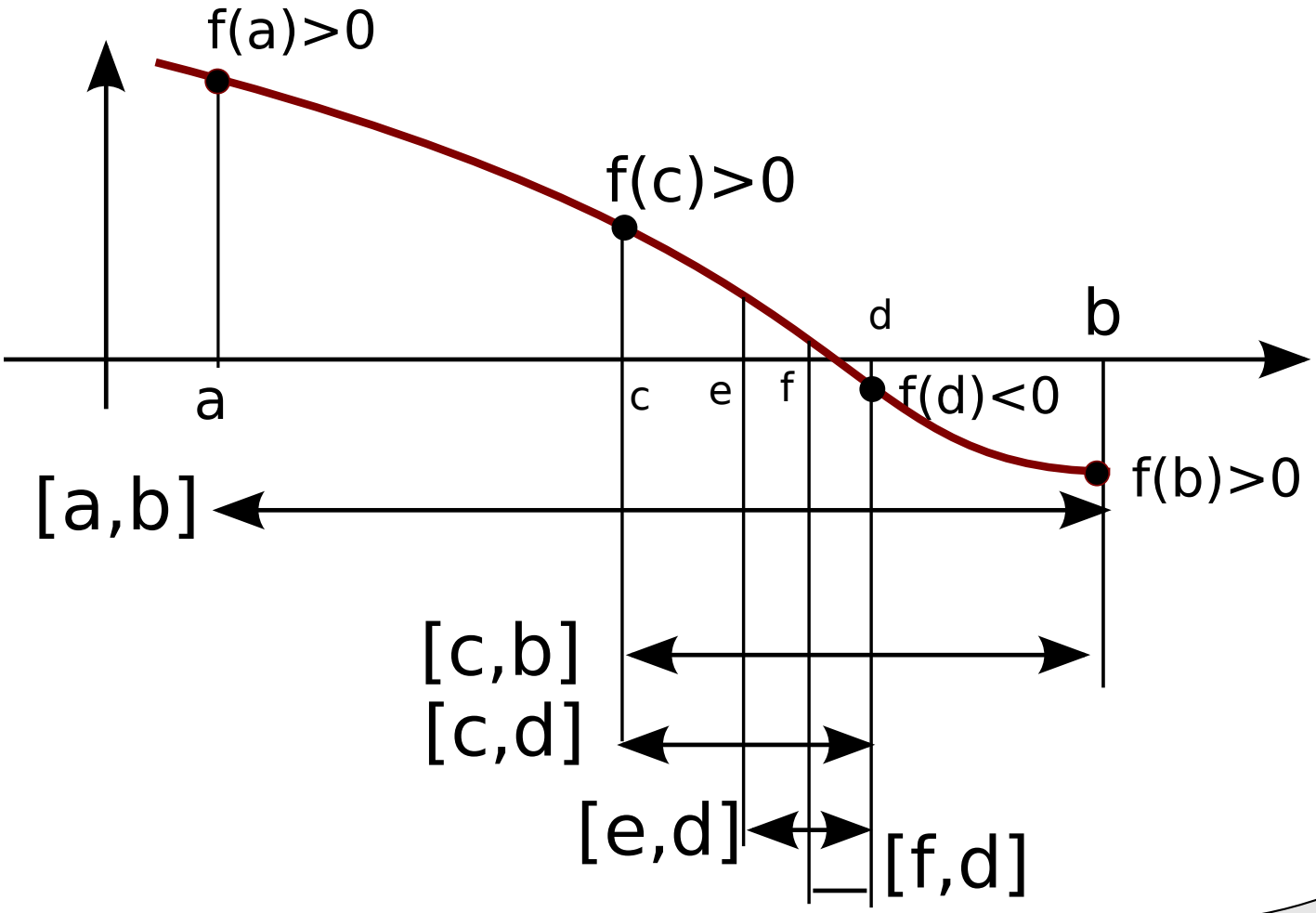
# Dichotomie



# Dichotomie



# Dichotomie



# Dichotomie

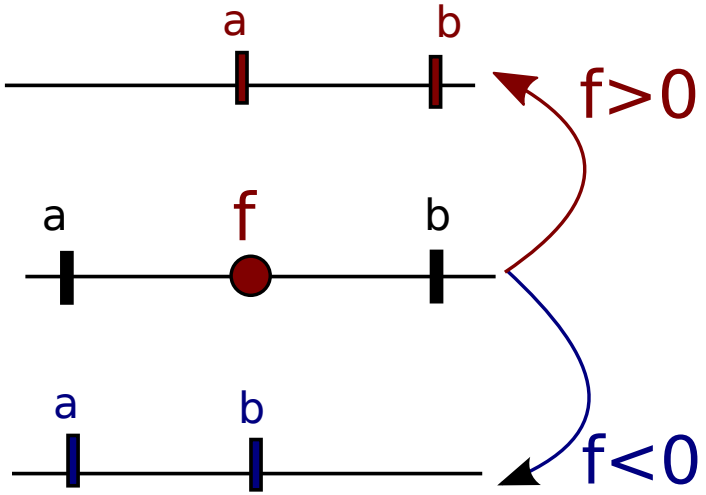
## Algorithme

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

**Tant que** |b-a|>erreur_max

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

**Si** f(c)>0
[a,b] <- [c,b]
**Sinon**
[a,b] <- [a,c]



# Dichotomie

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

# Arguments en Python

# Typage fort

Python est un langage "fortement" typé (contrairement au C !)

```
def func(a):
    s=type(a)
    print(s)

    return [a,5]


func("cheval")
func(8)
func(1.4)
func(True)
func([4,7])
func([])
```

Le type est connu à tout instant

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'list'>
<class 'list'>
```

# Type

```
def func(a):
    if isinstance(a,str):
        return "string"
    elif isinstance(a,float):
        print(a+4)
    else:
        return [4,5]

func("cheval")
func(4.5)
```

rem.
Retour variable

isinstance : comparaison du type

# ID

```
a=[4,7,8];
b=[4,7,8];
c=a;

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

id ~ adresse

=> Mais on n'accède pas à l'objet à partir de l'id!

# Fonctions arguments

Les fonctions peuvent être passées en tant qu'argument

```
def affichage_simple(T):
    print(T)

def affichage_debug(T):
    print("type : ",type(T))
    print("id : ",id(T))
    print("length : ",len(T))
    print(T)

def squared(v,afficher):
    s=0
    v2=[vk**2 for vk in v]
    afficher(v2)

v=range(-5,15,4)
aff=affichage_debug

squared(v,aff)
```

variable

ex.
Optimisation/
Intégration
numérique