Thème 1 - TP 1 : fonction de $\mathbb{R}$ dans $\mathbb{R}$, expression algébrique, représentation graphique, image réciproque d'un élément par la fonction.

Author **FX D**

Date **2017-05-06T14:30:43**

Project **a56d8802-5740-429a-b5da-e7eb8dc5bb82**

Location **1.sagews**

Original file **1.sagews**

Thème 1 - TP 1 : fonction de $\mathbb{R}$ dans $\mathbb{R}$, expression algébrique, représentation graphique, image réciproque d'un élément par la fonction.

Application à la densité de la loi normale et ses quantiles

Sujet

On se souvient que la densité de la loi normale centrée est de la forme $f(x) = a\exp(-bx^2)$ où a et b sont tels que l'intégrale sur $\mathbb{R}$ vaut 1. Trouver la relation entre a et b puis exprimer a et b en fonction l'écart type ; donner l'expression de f avec comme paramètre l'écart type. Comment obtient on la densité associée à une espérance m au lieu de 0 ?

Dessiner le graphe de f pour différentes valeurs de l'espérance et de l'écart type.

Trouver a tel que l'intégrale de f sur [-a,a] vaut 0.95 (Cf intervalle de confiance à 95%).

Au passage : recherche de documentation, éléments de programmation en Python : déclaration - définition de variables, affectation, expressions algébriques, valeur numérique, substitution dans une expression, fonctions (voir cette page (en))

Un corrigé

Distinguer les difficultés de syntaxe (syntaxePython), de programmation (méthodes), de documentations, algorithmiques, mathématiques : faire la part des choses

1. Approche élémentaire (le moins d'astuce de programmation que possible)

```
#faire une recherche sur "sagemath intégration", menu Calculus de la feuille interactive SageMathCloud, ...
integrate(exp(-x^2),x,0,1)
integrate(exp(-x^2),x)
I=integrate(exp(-x^2),x,-infinity,infinity);I
```

```
1/2*sqrt(pi)*erf(1)
1/2*sqrt(pi)*erf(x)
sqrt(pi)
```

```
I.n?
#commande suivi de ? pour une documentation sur la commande. 1ères lettres de la commande suivi de <tab> pour listes des commandes de préfixe do
#recherche sur les mots clef "sagemath valeur numérique"
#numerical_approx? erf?
```

```
File: /projects/sage/sage-7.5/src/sage/structure/element.pyx
Signature : I.n(self, prec=None, digits=None, algorithm=None)
Docstring :
Alias for "numerical_approx()".

EXAMPLES:

   sage: (2/3).n()
   0.666666666666667
```

```
integrate(exp(-x^2),x,0,1).n(digits=3)
I.n(digits=3)
numerical_approx(I,digits=3)
```

```
0.747
1.77
1.77
```

```
reset() # supprime les variables déjà déclarées ou affectées ; la réexécution produira le même résultat
#L'intégrale comme fonction de paramètres :
var('a b');III=integrate(a*exp(-(x*b)^2),x,-infinity,infinity,hold=true);III
#sans déclaration des paramètres autres que x, la commande donnerait une erreur. Cf var? , show_identifiers()
#hold=true pour que III renvoie l'expression littérale et non celle obtenue après calcul
#sans hold=true, le calcul produit une erreur, invite à précéder le calcul de assume(b>0)
#
III.parent() #donne la structure dont III est un élément, III n'est pas une fonction
```

```
#
#Comparer avec :
reset();print
II(a,b)=integrate(a*exp(-(x*b)^2),x,-infinity,infinity,hold=true)
#cette commande déclare II comme fonction et ses arguments a,b
show_identifiers()
II
II.parent()
II(1,1)#hold=true est oublié, comparer avec integrate(exp(-(x)^2),x,-infinity,infinity,hold=true)
```

```
(a, b)
integrate(a*e^(-b^2*x^2), x, -Infinity, +Infinity)
Symbolic Ring

['a', 'II', 'b']
(a, b) |--> integrate(a*e^(-b^2*x^2), x, -Infinity, +Infinity)
Callable function ring with arguments (a, b)
sqrt(pi)
```

```
show(II) #Visualisation LaTeX de l'expression II
```

$$(a,b) \mapsto \int_{-\infty}^{+\infty} a e^{\left(-b^{2} x^{2}\right)}\,dx$$

```
assume(b>0);simplify(II)
assumptions()
#sans assume(b>0) la commande simplify(II) produit une erreur, le message d'erreur invite à utiliser simplify
#assume(b>0) reste actif pour les cellules suivantes. Pour revenir en arrière forget(b>0), pour tout oublier forget()
#cf assume? assuptions? forget?
```

```
(a, b) |--> sqrt(pi)*a/b
[b > 0]
```

```
#relation entre a et b pour que II(a,b)=1, cf recherche sur mot clef sagemath solution équation
#http://doc.sagemath.org/html/fr/tutorial/tour_algebra.html
solve(II(a,b)==1,a) #assume(b>0) implicite, sans quoi une erreur est produite
```

```
[a == b/sqrt(pi)]
```

```
#densité gaussienne centrée
a = b/sqrt(pi);f(x)=a*e^(-b^2*x^2)
f
integrate(f(x),x,-infinity,infinity)
```

```
x |--> b*e^(-b^2*x^2)/sqrt(pi)
1
```

```
#Espérance
E1=integrate(x*f(x),x,-infinity, infinity);E1
```

```
0
```

```
#Variance
E2=integrate(x^2*f(x),x,-infinity, infinity);V=E2-E1^2;V
show(V)
```

```
1/2/b^2
```

$$\frac{1}{2\,b^{2}}$$

```
#b en fonction de l'écart-type puis expression de f
var('s');solve(V==s^2,b)
```

```
s
[b == -1/2*sqrt(2)/s, b == 1/2*sqrt(2)/s]
```

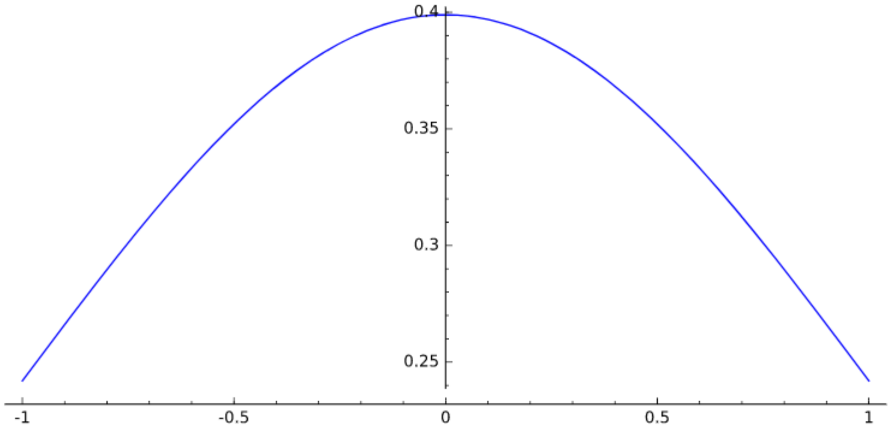
```
#substitution à la main dans l'expression de f
b = 1/2*sqrt(2)/s
f(x)=b*e^(-b^2*x^2)/sqrt(pi)
f
show(f)
```

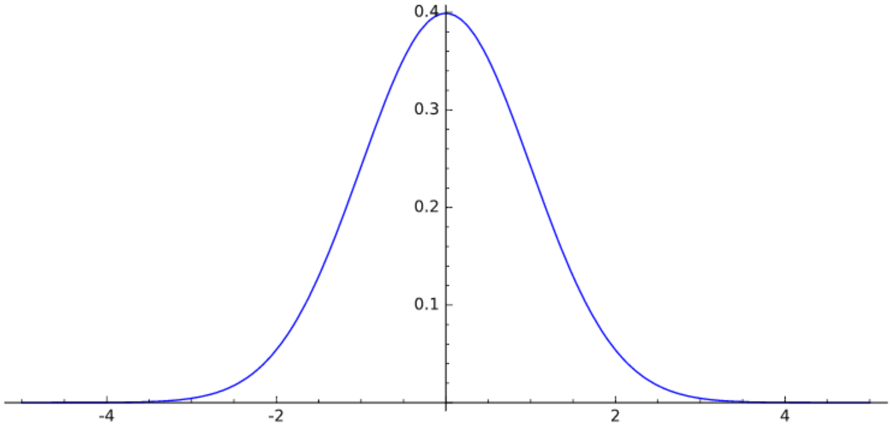
```
x |--> 1/2*sqrt(2)*e^(-1/2*x^2/s^2)/(sqrt(pi)*s)
```

$$x \mapsto \frac{\sqrt{2} e^{\left(-\frac{x^{2}}{2\,s^{2}}\right)}}{2\,\sqrt{\pi} s}$$

```
#Représentation graphique
#s=1;plot(f) produit une erreur : s=1 n'est pas pris en compte dans l'évaluation de f. La bonne méthode est f(s=1) cf "sagemath substitution exp
s=1;f(1)
f(s=1)
g(x)=f(s=1)
g
plot(g)
#google "sagemath plot" ou plot?
plot(g,-5,5)
```

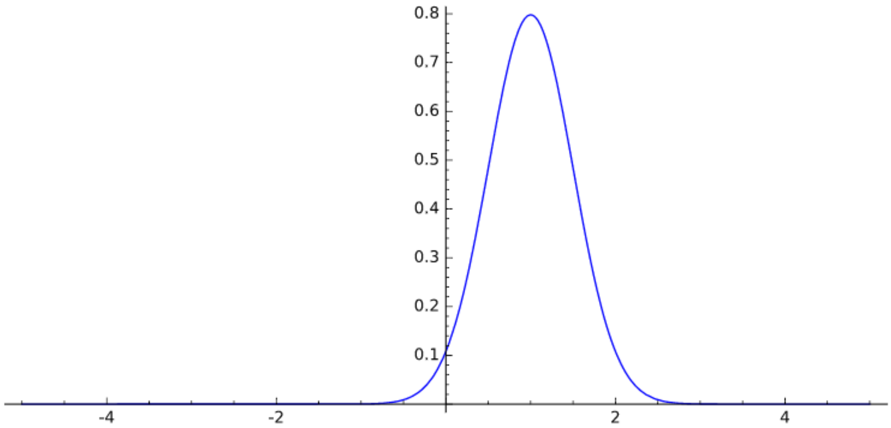
```
1/2*sqrt(2)*e^(-1/2/s^2)/(sqrt(pi)*s)
1/2*sqrt(2)*e^(-1/2*x^2)/sqrt(pi)
x |--> 1/2*sqrt(2)*e^(-1/2*x^2)/sqrt(pi)
```






```
#densité gaussienne centrée en 1, d'écart type 1/2
g(x)=f(s=1/2,x=x-1) #drole de syntaxe !
g
plot(g,-5,5)
```

```
x |--> sqrt(2)*e^(-2*(x - 1)^2)/sqrt(pi)
```




```
#a en fonction de s tel que l'intégrale de f sur [-a,a] vaut 0.95
var('a');integrate(f(x),x,-a,a)
```

```
a
erf(1/2*sqrt(2)*a/s)
```

```
var('s') #réinitialisation de s
solve(erf(1/2*sqrt(2)*a/s)==0.95,a)
```

```
s
[a == sqrt(2)*s*inverse_erf(19/20)]
```

```
#a est proportionnel à s, ça se voyait déjà sur l'équation
#valeur numérique de a pour s=1 ?
s=1;(sqrt(2)*s*inverse_erf(19/20)).n()
```

```
Error in lines 1-1
Traceback (most recent call last):
  File "/projects/sage/sage-7.5/local/lib/python2.7/site-packages/smc_sagews/sage_server.py", line 995, in execute
    exec compile(block+'\n', '', 'single') in namespace, locals
  File "", line 1, in
NameError: name 'inverse_erf' is not defined

*** WARNING: Code contains non-ascii characters ***
```

```
#inverse_erf donné dans une réponse par Sage n'est pas (en mai 2017) un objet de Sage
#voir à ce sujet https://ask.sagemath.org/question/8118/evaluating-inverse-erf/
#Une version numérique de inverse_erf peut être importée de la librairie Python Scipy, cf recherche sur mots clef "python inverse erf"
from scipy.special import erfinv as inverse_erf
inverse_erf?
#On revient en arrière par la commande
#del inverse_erf
```

```
File: /projects/sage/sage-7.5/local/lib/python2.7/site-packages/scipy/special/basic.py
Signature : inverse_erf(y)
Docstring :
Inverse function for erf.
```

```
s=1;(sqrt(2)*s*inverse_erf(19/20)).n()
```

```
1.95996398454005
```

2. Résolution numérique d'une équation

```
#Méthode implémentée dans Python, cf find_root?
#Recherche d'une valeur approchée de a entre 0 et 4
find_root(erf(1/2*sqrt(2)*a)==0.95,0,4)
```

```
1.959963984540053
```

```
#Algorithme de dichotomie : recherche d'un 0 de la fonction h entre a et b à la précision prec si h change de signe entre a et b
def dicho(h,a,b,prec):
    if h(a)*h(b)>0:
        print('peut être pas de solution')
    else:
        while b-a>prec:
            i=(a+b)/2
            if h(i)*h(b)>0:b=i
            else:a=i
    return(a)

h(x)=erf(1/2*sqrt(2)*x)-0.95
dicho(h,0,4,0.001).n() #seules les 3 premières décimales après la virgule sont pertinentes !
```

```
1.95996093750000
```

```
#Vérification
show(integrate(f(s=1),x,-1.96.n(digits=3),1.96.n(digits=3),hold=true),'=',integrate(f(s=1),x,-1.96,1.96).n())
```

$$\int_{-1.96}^{1.96} \frac{\sqrt{2}e^{\left(-\frac{1}{2}\,x^{2}\right)}}{2\,\sqrt{\pi}}\,dx = 0.950004209703559$$

3. Fonction de répartition

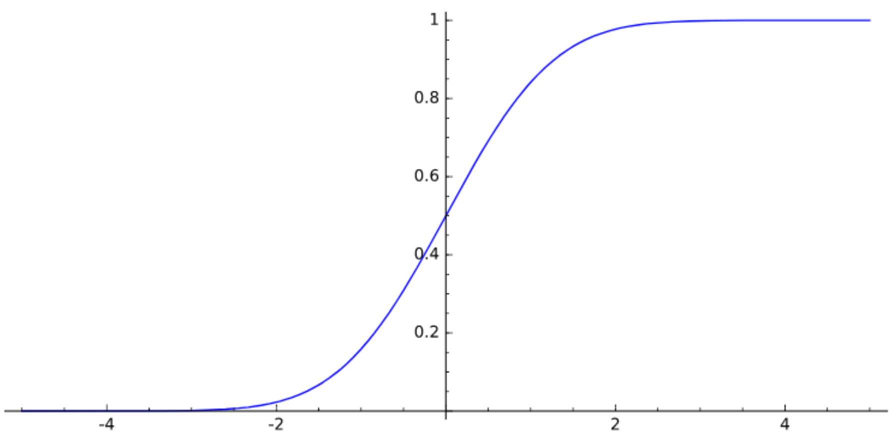
```
#Fonction de répartition de la densité gaussienne centrée réduite (s=1)
F(t)=integrate(f(s=1),x,-infinity,t);F
```

```
t |--> 1/4*sqrt(2)*(sqrt(2)*sqrt(pi)*erf(1/2*sqrt(2)*t) + sqrt(2)*sqrt(pi))/sqrt(pi)
```

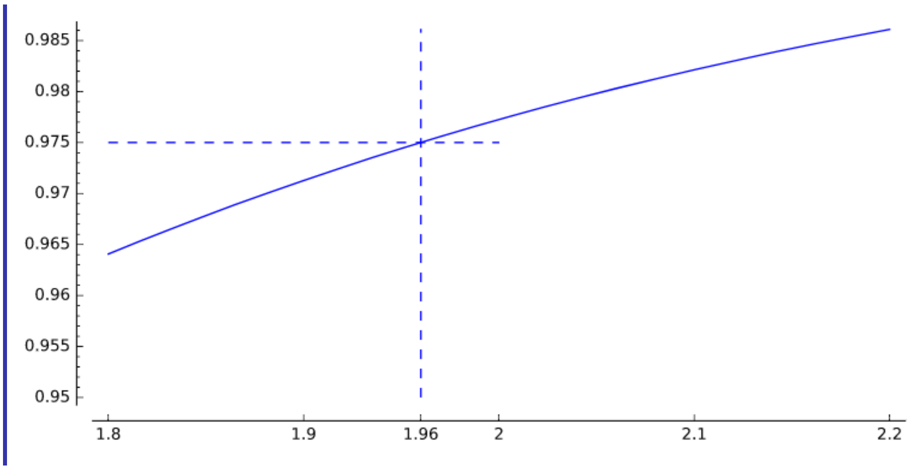
```
#Méthode simplify disponible pour F (faire F. suivi de <tab>)
F.simplify_full() #bizarement on obtient F(t) plutôt que F
F(t)=F.simplify_full();F
```

```
1/2*erf(1/2*sqrt(2)*t) + 1/2
t |--> 1/2*erf(1/2*sqrt(2)*t) + 1/2
```

```
plot(F,-5,5)
```




```
#On cherche a tel que F(a)-F(-a)=0.95 soit F(a)=0.975, d'abord sur le dessin
#1.96 est choisi "à la main" et on vérifie par le tracé des lignes en pointillés que c'est une bonne approximation
#ticks en option : voir http://doc.sagemath.org/html/en/reference/plotting/sage/plot/plot.html
plot(F,1.8,2.2)+line([(1.8,0.975),(2,0.975)],linestyle='--',ticks=[[1.8,1.9,1.96,2,2.1,2.2],None])+line([(1.96,0.95),(1.96,F(2.2))],linestyle='--')
```




```
#Méthode numérique avec find_root
find_root(F(x)==0.975,0,4)
```

```
1.959963984540053
```

```
#Méthode formelle
solve(F(x)==0.975,x) #même solution que précédemment, même pb avec inverse_erf
```

```
[x == sqrt(2)*inverse_erf(19/20)]
```

4. Programmation plus efficace. Qu'est qui change par rapport à la section 1. ?

```
show_identifiers()
reset();print;show_identifiers()
```

```
['g', 'dicho', 'V', 'F', 'h', 's', 't', 'inverse_erf', 'a', 'E1', 'E2', 'II', 'b', 'f']

[]
```

```
var('a','b')
f(x)=a*exp(-(b*x)^2);f
var('m')
assume(b != 0);I=integrate(f(x-m),x,-infinity,infinity);I #sans assume(b != 0) une erreur est produite dont le texte invite à une telle commande
```

```
(a, b)
x |--> a*e^(-b^2*x^2)
m
sqrt(pi)*a/sqrt(b^2)
```

```
assume(b>0);simplify(I)
```

```
sqrt(pi)*a/b
```

```
S=solve(I==1,a);S
```

```
[a == b/sqrt(pi)]
```

```
f=f.subs(S[0]);f
#alternative : f=f(a=S[0].rhs())
```

```
x |--> b*e^(-b^2*x^2)/sqrt(pi)
```

```
E1=integrate(x*f(x-m),x,-infinity,infinity);E1 #espérance
E2=integrate(x^2*f(x-m),x,-infinity,infinity);V=E2-E1^2;V #variance
V=V.simplify_full();V #V ne devrait pas dépendre de m !
show('V = ',V)
```

```
m
-m^2 + 1/2*(2*sqrt(pi)*m^2/b + sqrt(pi)/b^3)*b/sqrt(pi)
1/2/b^2
```

V = $\frac{1}{2\,b^{2}}$

```
var('s');B=solve(s^2==V,b);B
```

```
s
[b == -1/2*sqrt(2)/s, b == 1/2*sqrt(2)/s]
```

```
f=f.subs(B[1]);show('f_s (x) = ',f)
```

f_s (x) = $x \ {\mapsto}\ \frac{\sqrt{2} e^{\left(-\frac{x^{2}}{2 \, s^{2}}\right)}}{2 \, \sqrt{\pi} s}$

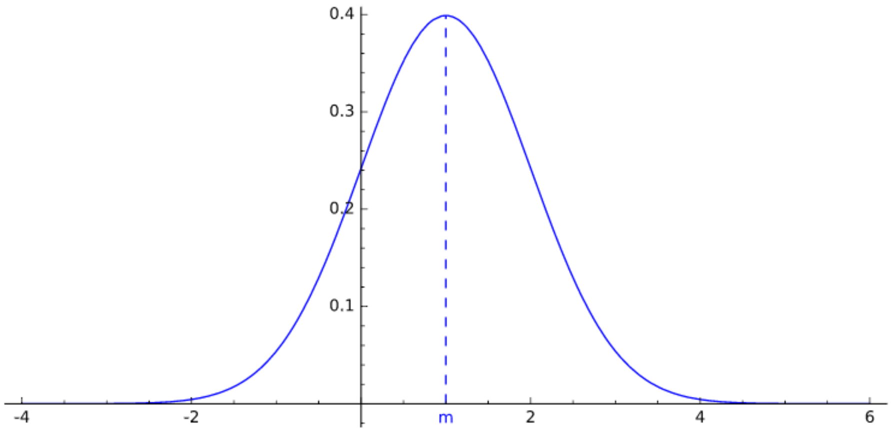
```
la='$'+latex(f)+'$'
html('<h3><div align="center"> $f_s(x)\ =\ $+la+'</div></h3>')
```

$$f_s(x)\ =\ x \ {\mapsto}\ \frac{\sqrt{2} e^{\left(-\frac{x^{2}}{2 \, s^{2}}\right)}}{2 \, \sqrt{\pi} s}$$

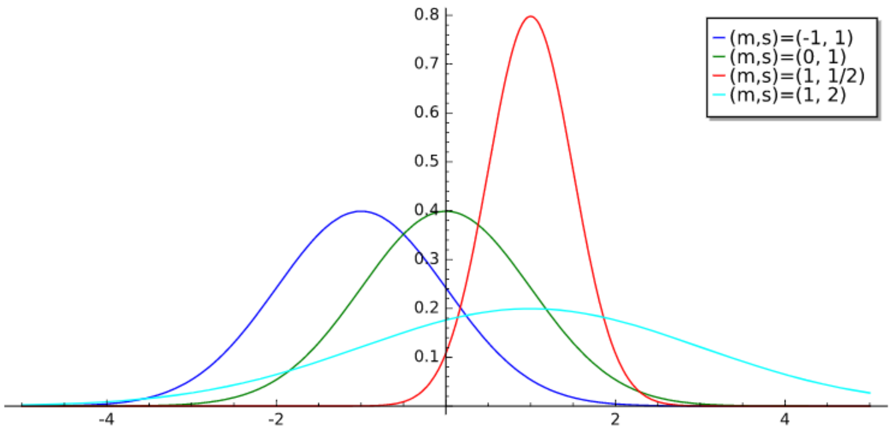
```
#densité gaussienne centrée en m
g(m,s,x)=f(x-m)
show(g)
```

$$(m,s,x) \mapsto \frac{\sqrt{2}e^{\left(-\frac{(m-x)^2}{2\,s^2}\right)}}{2\,\sqrt{\pi}s}$$

```
#Représentation graphique, cf "sagemath graphique", plot?
m=1;s=1
plot(g(m,s,x),x,-4,6)+line([(m,0),(m,g(m,s,m))],linestyle='--')+text("m",(m,-0.015))
```




```
c = ['blue','green','red','cyan','magenta','yellow','black']
m=[-1,0,1,1];s=[1,1,1/2,2]
p=sum(plot(g(m[k-1],s[k-1],x),x,-5,5, color=c[k-1],legend_label='(m,s)='+str((m[k-1],s[k-1]))) for k in (1..4))
show(p)
```




```
#Recherche d'un quantile pour s=1
#Méthode formelle avec la fonction de répartition
from scipy.special import erfinv as inverse_erf
F(t)=integrate(g(0,1,x),x,-infinity,t)
solq=solve(F(x)==1.95/2,x);solq
x.subs(solq) #la définition scipy de inverse_erf n'affecte pas la méthode solve
#x.subs(solq).n() produit une erreur ; on recopie à la main le texte de la solution ; peut on le faire par une commande à partir de str(solq[0])
x.subs(x == sqrt(2)*inverse_erf(19/20)).n(digits=3)
```

```
[x == sqrt(2)*inverse_erf(19/20)]
sqrt(2)*inverse_erf(19/20)
1.96
```

```
#test
%python
import platform
platform.python_version()
1/2 #le résultat est un entier dans python 2.* ! cf "python division"
1./2
2^3 #Cf la section 9.9.1 de https://docs.python.org/2/library/operator.html
2**3
```

```
'2.7.13'
0
0.5
1
8
```

```
%python3
import platform
print(platform.python_version())
print(1/2)
```

```
3.5.3
0.5
```

```
#test dans sage
from scipy.special import erfinv as inverse_erf
eval('1/2')
eval('1./2')
sage_eval('1/2')
#sage_eval('inverse_erf(1/2)') produit une erreur : inverse_erf pas reconnu
eval('inverse_erf(1/2)')
eval('inverse_erf(1./2)')
```

```
0
0.5
1/2
0.0
0.47693627620446982
```

```
eval(str(solq[0].rhs()).replace('/','*1.0/')).n()
```

```
1.95996398454005
```

```
#Algorithme de dichotomie
def dicho(h,a,b,prec):
    if h(a)*h(b)>0:
        print('peut être pas de solution')
    else:
        while b-a>prec:
            i=(a+b)/2
            if h(i)*h(b)>0:b=i
            else:a=i
    return(a)
```

```
dicho(F-0.975,0,4,0.001).n() #seules les 3 premières décimales après la virgule sont pertinentes !
```

```
1.95996093750000
```

Compléments : variables, expressions, fonctions, affectations dans Python

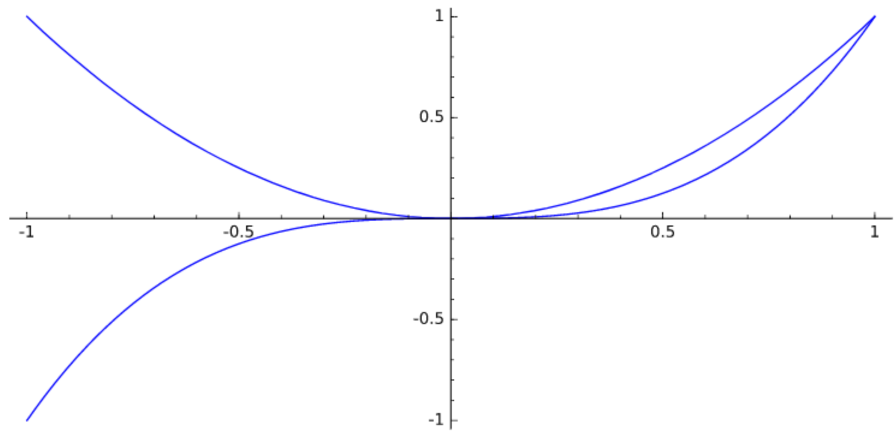
```
#Test
reset()
f=x;f #x est prédéclaré ; f=y produit une erreur si y n'est pas déclaré au préalable
x=1;f #pas rétroactif
f=x;f #x est remplacé par sa valeur
f(x)=x;f;x#x est réinitialisé !
print
f=x;f
f.parent()
f.variables()
f(x=1) #substitution de x par 1 dans l'expression f
show_identifiers() #x est omis par convention
print
g=f;g
x=1
diff(f,x)
x=var('x');diff(f)#avec x=1 produit une erreur
integrate(f)
var('y');integrate(f(x=y),y)#avec y=1 produit une erreur
print
g(x)=f;print g,',', g.parent(),',',g.variables()
#g(y)=f.subs(x=y);g
g.diff()
diff(g(y),y)
integrate(g)
integrate(g(y),y)
```

```
x
x
1
x |--> x
x

x
Symbolic Ring
(x,)
1
['f']

x
1
1
1/2*x^2
y
1/2*y^2

x |--> x , Callable function ring with argument x , (x,)
x |--> 1
1
x |--> 1/2*x^2
1/2*y^2
```

```
reset()
h=lambda x:x^2;h;h(2);h(x);diff(h(x),x);integrate(h(x),x)
#mais diff(h) produit une erreur, de même que h.variables()
```

```
print
def k(x):return x^3
k;k(x);diff(k(x),x)
show_identifiers()
plot(h)+plot(k)
```

```
 at 0x7f10bb28db18>
4
x^2
2*x
1/3*x^3


x^3
3*x^2
['k', 'h']
```




```
reset()
#y n'est pas défini, une expression contenant y produit une erreur, mais pas une affectation
g(y)=x;g;g(1) #déclare g et y
show_identifiers()
g.parent()
```

```
y |--> x
x
['g', 'y']
Callable function ring with argument y
```

```
#variables locales dans def, affectation globale
reset()
x=2
def g(y):
    z=y
    y=1
    x=1
    return(z)
show_identifiers()
g(x)
x
```

```
['g', 'x']
2
2
```

```
str(g(y))
```

```
'x'
```

```
import ast
formula = 'integrate(y+a^2,a,0,1)'
names = [
    node.id for node in ast.walk(ast.parse(formula))
    if isinstance(node, ast.Name)
]
names
```

```
['integrate', 'a', 'y', 'a']
```

```
del ast
```