Exercices Python

Fonctions
Listes
Courbes

Fonctions

Soit p=[a,b,c] le vecteur correspondant aux coefficients du polynome: ax^2+bx+c

Construire une fonction qui recoit en argument p et renvoie les 2 racines dans R si elles existent.

Fonctions

Soit p=[a,b,c] le vecteur correspondant aux coefficients du polynome: ax^2+bx+c

Construire une fonction qui recoit en argument p et renvoie les 2 racines dans R si elles existent.

```
from math import sqrt

def racine(p):
    a,b,c=p
    delta=b*b-4*a*c

    epsilon=1e-6
    if delta<0:
        print("Pas de racines reelles")
    elif delta<epsilon:
        return -b/(2*a)
    else:
        return [(-b+sqrt(delta))/(2*a),(-b-sqrt(delta))/(2*a)]

r=racine([1,8,2])
print(r)
```

Fonctions

Soit p=[a,b,c] le vecteur correspondant aux coefficients du polynome: ax^2+bx+c

Construire une fonction qui recoit en argument p et renvoie les 2 racines dans C.

Rem: a+1j*b représente un nombre complexe

Fonctions

Soit p=[a,b,c] le vecteur correspondant aux coefficients du polynome: ax^2+bx+c

Construire une fonction qui recoit en argument p et renvoie les 2 racines dans C.

Rem: a+1j*b représente un nombre complexe

```
from math import sqrt

def racine(p):
    a,b,c=p
    delta=b*b-4*a*c

    if delta<0:
      return [(-b+1j*sqrt(-delta))/(2*a),(-b-1j*sqrt(-delta))/(2*a)]
    else:
      return [(-b+sqrt(delta))/(2*a),(-b-sqrt(delta))/(2*a)]

r=racine([6,1,2])
print(r)
```

Entree utilisateur

Définir un nombre a entier (aleatoire) entre 0 et N.

```
import random
random.random()
```

Demander à l'utilisateur de saisir un nombre entre 0 et N.
Indiquez si ce nombre est plus grand ou plus petit que a.
Recommencez l'opération jusqu'à avoir trouvé le nombre correspondant.

```
input("saisir valeur : ")
```

Entree utilisateur

Définir un nombre a entier (aleatoire) entre 0 et N.

```
import random
random.random()
```

Demander à l'utilisateur de saisir un nombre entre 0 et N.
Indiquez si ce nombre est plus grand ou plus petit que a.
Recommencez l'opération jusqu'à avoir trouvé le nombre correspondant.

```
input("saisir valeur : ")
```

```
import random

r=int(random.random()*500)

x=0
while x!=r:
    x=int(input("valeur : "))
    if x>r:
        print("Plus petit")
    elif x<r:
        print("Plus grand")
print("Gagne")
```

Tableaux

Soit V=[10.5,7.2,8.4,14.1,2.5,12.1,6.5,19.5]
Calculer la moyenne et l'écart type de V.

Tableaux

Soit V=[10.5,7.2,8.4,14.1,2.5,12.1,6.5,19.5]
Calculer la moyenne et l'écart type de V.

```
from math import sqrt

V=[10.5,7.2,8.4,14.1,2.5,12.1,6.5,19.5]

avg=sum(V)/len(V)
std=sqrt(sum([(vk-avg)**2 for vk in V])/len(V))
print(avg,std)
```

Application - Intégration Monte Carlo

- Soit le domaine 3D de l'espace $D=[-1,1]^3$
- On note V le volume de D
- On choisi N positions tirés aléatoirement dans le domaine D
- On appelle H le nombre de positions situés à l'intérieur de la sphère de rayon 1
- Calculer $I=V\ H\ /N$, et observer que I converge vers le volume de la sphère de rayon 1

```
import random

random.uniform(a,b)
```

Application - Intégration Monte Carlo

- Soit le domaine 3D de l'espace $D=[-1,1]^3$
- On note V le volume de D
- On choisi N positions tirés aléatoirement dans le domaine D
- On appelle H le nombre de positions situés à l'intérieur de la sphère de rayon 1

```
import random

random.uniform(a,b)
```

- Calculer I=V H /N, et observer que I converge vers le volume de la sphère de rayon 1

```
import random


V=2*2*2
N=50000

H=0
for k in range(N):
    p=[random.uniform(-1,1),random.uniform(-1,1),random.uniform(-1,1)]
    if p[0]*p[0]+p[1]*p[1]+p[2]*p[2]<1:
        H+=1

I=V*H/N
print(I)
```

Equation de récurrence

Soit la relation de récurrence suivante

$$x^{k+2} = 2x^{k+1} - 3x^{k} \quad \text{pour } k >= 0$$

et $x^{0}=1$, $x^{1}=2$

Calculer et afficher les 10 premiers termes de cette relation

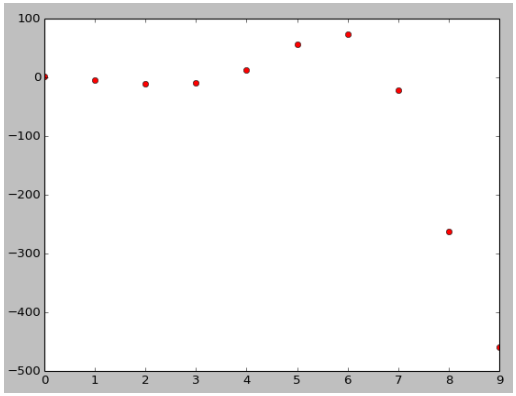
Equation de récurrence

Soit la relation de récurrence suivante

$x^{k+2} = 2x^{k+1} - 3x^{k}$ pour k >= 0

et $x^{0}=1$, $x^{1}=2$

Calculer et afficher les 10 premiers termes de cette relation



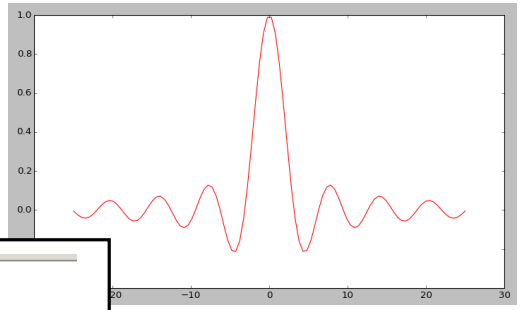

```
import matplotlib.pyplot as plt
import math

x0=1
x1=2

T=[]
for k in range(10):
    x=2*x1-3*x0
    x0=x1
    x1=x

    T.append(x)

plt.plot(T,"ro")

plt.show()
```

Affichage 1D

Tracez la fonction sin(x)/x

Affichage 1D

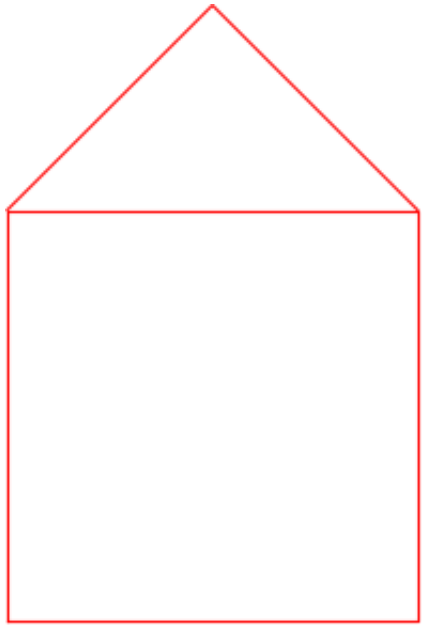
Tracez la fonction sin(x)/x




```
import matplotlib.pyplot as plt
import math

N=100
t=[k/(N-1) for k in range(N)]
x=[25*2*(t[k]-0.5) for k in range(N)]

epsilon=1e-6
x=[25*2*(t[k]-0.5) for k in range(N)]
f=[math.sin(x[k])/(x[k]+epsilon) for k in range(N)]

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

Affichage 2D : figure géométrique

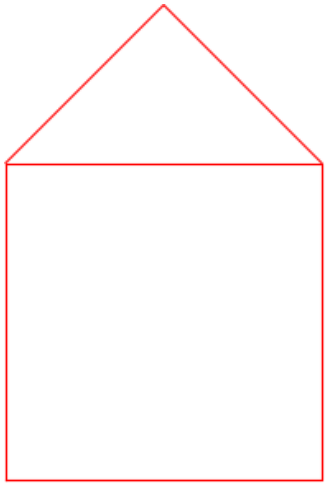
Tracez la figure représentant cette maison



Affichage 2D : figure géométrique

Tracez la figure représentant cette maison

```
import matplotlib.pyplot as plt
import math

p0=[0,0]
p1=[1,0]
p2=[1,1]
p3=[0,1]
p4=[0.5,1.5]

carre=[p0,p1,p2,p3,p0]
toit=[p3,p4,p2]

plt.plot([c[0] for c in carre],[c[1] for c in carre],"r")
plt.plot([t[0] for t in toit],[t[1] for t in toit],"r")
plt.axis("equal")
plt.axis([-0.5,1.5,-0.2,1.8])
plt.show()
```



Bézier

Une courbe de Bézier de degré 3 est donnée par

$$B(t)=(1-t)^3 A + 3(1-t)^2 t B + 3(1-t)t^2 C + t^3 D$$

Avec (A,B,C,D), le polygone de contrôle de la courbe de Bézier, et t variant entre 0 et 1.

Construire la fonction qui calcule B(t) pout t donné

Afficher une courbe de Bézier en calculant N échantillons

Bézier

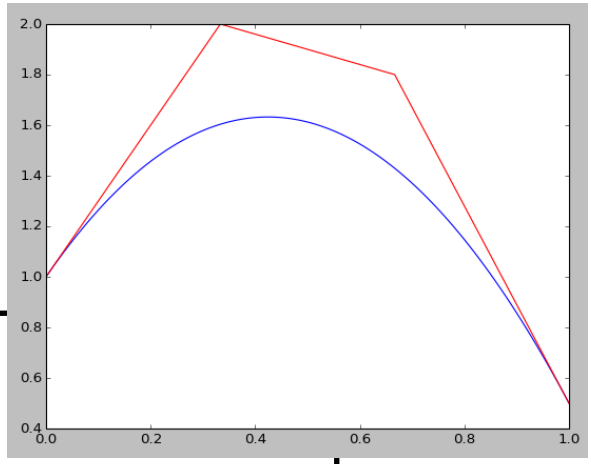
Une courbe de Bézier de degré 3 est donnée par

$B(t)=(1-t)^3 A + 3(1-t)^2 t B + 3(1-t)t^2 C + t^3 D$

Avec (A,B,C,D), le polygone de contrôle de la courbe de Bézier, et t variant entre 0 et 1.

Construire la fonction qui calcule B(t) pout t donné

Afficher une courbe de Bézier en calculant N échantillons




```
import matplotlib.pyplot as plt

A=1
B=2
C=1.8
D=0.5

def bezier(t):
    return A*(1-t)**3+3*B*(1-t)**2*t+3*C*(1-t)*t**2+D*t**3

N=100
x=[k/(N-1) for k in range(N)]
y=[bezier(xk) for xk in x]

plt.plot(x,y)
plt.plot([0,1/3,2/3,1],[A,B,C,D],'r')

plt.show()
```

Affichage courbe 2D

Tracer la fonction

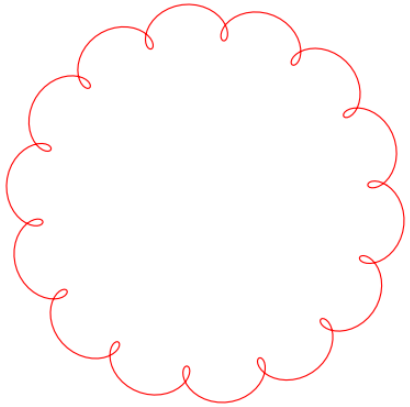
```
x(t)=cos(2 pi t)-0.1 sin(15 pi t)
y(t)=sin(2 pi t)+0.1 cos(15 pi t)
```

Affichage courbe 2D

Tracer la fonction

$x(t)=\cos(2\ \pi\ t)-0.1\ \sin(15\ \pi\ t)$

$y(t)=\sin(2\ \pi\ t)+0.1\ \cos(15\ \pi\ t)$



```
import matplotlib.pyplot as plt
import math

N=400
t=[2*math.pi*k/(N-1) for k in range(N)]

x=[math.cos(t[k])-0.1*math.sin(15*t[k]) for k in range(N)]
y=[math.sin(t[k])+0.1*math.cos(15*t[k]) for k in range(N)]

plt.plot(x,y,"r")
plt.axis("equal")
plt.show()
```

Equation différentielle

Soit l'équation différentielle ordinaire donnée par

$$x'(t) = y \qquad \text{avec } x(0)=x_0$$
$$y'(t) = -x \qquad y(0)=y_0$$

On peut montrer qu'un schéma numérique d'intégration (Runge-Kutta) permet d'approximer la trajectoire (x,y) avec la relation suivante:

$$x^{k+1} = (1 - dt^2/2)x^k + dt\, y^k$$
$$y^{k+1} = (1 - dt^2/2)y^k - dt\, x^k$$

Calculer N itération de cette récurrence
quelle est la trajectoire obtenue?

Equation différentielle

Soit l'équation différentielle ordinaire donnée par

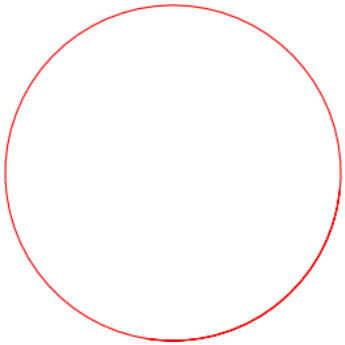
$x'(t) = y$ avec $x(0)=x_0$

$y'(t) = -x$ $y(0)=y_0$

On peut montrer qu'un schéma numérique d'intégration (Runge-Kutta) permet d'approximer la trajectoire (x,y) avec la relation suivante:

$$x^{k+1} = (1 - dt^2/2)x^k + dt\ y^k$$
$$y^{k+1} = (1 - dt^2/2)y^k - dt\ x^k$$

Calculer N itération de cette récurrence
quelle est la trajectoire obtenue?



```python
import matplotlib.pyplot as plt
import math

x0,y0=1,0
dt=0.1

Tx,Ty=[],[]
for k in range(80):
    x=(1-dt*dt/2)*x0+dt*y0
    y=(1-dt*dt/2)*y0-dt*x0

    x0,y0=x,y

    Tx.append(x)
    Ty.append(y)

plt.plot(Tx,Ty,"r")
plt.axis("equal")
plt.show()
```

Intégration d'équation différentielle

Soit une particule de masse m (on peut considérer m=1)

Cette particule est lancée avec une vitesse initiale v0
depuis une position initiale x0

L'équation du mouvement de cette particule est donnée par le système

$$v'(t)=g$$
$$x'(t)=v(t)$$

Soit, la relation de récurence

$$\begin{cases} v_x^{k+1}=v_x^k \\ v_y^{k+1}=v_y^k-||g|| \\ x^{k+1}=v_x^k \\ y^{k+1}=v_y^k \end{cases}$$

Calculer et afficher la trajectoire pour N itérations
Quelle trajectoire est attendue?

Intégration d'équation différentielle

Ajouter une force de frottement fluide du type
$F=-mu\ v^2$

La force F est orientée dans le sens opposé à la vitesse

Afficher les trajectoires obtenues pour différentes valeurs de mu

Intégration d'équation différentielle

Ajouter une force de frottement fluide du type
$F=-mu\ v^2$

La force F est orientée dans le sens opposé à la vitesse

Afficher les trajectoires obtenues pour différentes valeurs de mu

```
def compute_traj(mu):
    N=500

    delta_t=0.0008

    g=9.81
    x,y=0,0
    vx,vy=1,1

    traj_x=[]
    traj_y=[]

    for k in range(N):
        x=x+delta_t*vx
        y=y+delta_t*vy

        norm_v2=vx*vx+vy*vy
        norm_v=math.sqrt(norm_v2)

        vx_unit=vx/norm_v
        vy_unit=vy/norm_v

        vx=vx+delta_t*( 0-mu*norm_v2*vx_unit)
        vy=vy+delta_t*(-g-mu*norm_v2*vy_unit)

        traj_x.append(x)
        traj_y.append(y)

    return [traj_x,traj_y]
```

```
mu=5
traj_1=compute_traj(5)
traj_2=compute_traj(15)
traj_3=compute_traj(0.0)


plt.plot(traj_1[0],traj_1[1],"r")
plt.plot(traj_2[0],traj_2[1],"b")
plt.plot(traj_3[0],traj_3[1],"k")

plt.show()
```