1. Proposer une fonction `integrer(f,a,b,n)` permettant de donner une valeur approchée de $\int_a^b f$ par une méthode de rectangle de pas $(b-a)/n$.

Code informatique 1.

```
def integrer(f,a,b,n):
    rep=0
    for i in range(n):
        rep = rep + (b-a)/n * f(a+i*(b-a)/n)
    return(rep)
```

2. Connaissez vous une autre méthode de calcul de $\int_a^b f$ plus performante ? Si oui, modifier votre fonction `integrer(f,a,b,n)` en conséquence.
La méthode des trapèzes :

Code informatique 2.

```
def integrer2(f,a,b,n):
    rep=0
    for i in range(n):
        rep = rep + (b-a)/n*1/2*(f(a+i*(b-a)/n)+f(a+(i+1)*(b-a)/n))
    return(rep)
```

3. En déduire une fonction `coeff_fourier(u0,k,L,n)` qui renvoie le coefficient de Fourier a_k de la fonction `u0`.

Code informatique 3.

```
from math import *

def coeff_fourier(u0,k,L,n):
    rep = 0
    a=0
    b=L
    for i in range(n):
        valeur1 = sin(k*pi*(a+i*(b-a)/n)/L) * u0(a+i(b-a)/n)
        valeur2 = sin(k*pi*(a+(i+1)*(b-a)/n)/L)*u0(a+(i+1)*(b-a)/n)
        rep = rep + (b-a)/n*1/2*(valeur1 + valeur2)
    return(rep*2/L)
```

4. Écrivez la fonction `solution_corde(u0,L,N,A,n,x,t)` qui étant donné la condition initiale `u0` de notre problème calcule la somme partielle jusqu'au terme `N` de la série solution u au point (x,t).

Code informatique 4.

```
def solution_corde(u0,L,N,A,n,x,t):
    rep = 0
    for k in range(1,N+1):
        a = coeff_fourier(u0,k,L,n)
        rep = rep + a * sin(k*pi*x/L)*cos(A*k*pi*t/L)
    return(rep)
```

5. Déterminer en fonction de `n` la complexité de votre fonction `integrer(f,a,b,n)` en nombre d'opérations arithmétiques (sommes, produits...)
On compte le nombre de boucles : la complexité de la fonction integrer est $C_1(n)$ est en $n * O(1) = O(n)$

6. En déduire en fonction de `n` et de `N` la complexité de `solution_corde(u0,L,N,A,n,x,t)`.
On compte le nombre de boucles : la complexité de la fonction integrer est $C_2(n,N)$ est en $N * O(n) = O(n * N)$

7. Proposer un programme permettant de visualiser le graphe de $x \mapsto u(x,t)$ à un temps donné $t \in [0, L]$.
(on appellera les bibliothèques nécessaires).

Code informatique 5.

```
from matplotlib import pyplot as plt
import numpy as numpy

X = np.linspace(0,L,501) # la valeur 501 est à modifier à la demande
Y = [solution_corde(u0,L,N,A,n,x,t) for x in X]

plt.plot(X,Y)
plt.show()
```

```
SELECT *
	FROM Rg;
SELECT *
	FROM Depts;
SELECT LIBGEO,P09POP
	FROM Communes
	WHERE DEP=90;
SELECT LIBGEO,P09POP
	FROM Communes
	WHERE DEP=37 AND P09POP- P99POP > 1000;
SELECT LIBGEO,P09POP
	FROM Communes
	WHERE P09POP>200000;
SELECT LIBGEO,P09POP,SUPERF,DEP
	FROM Communes
	WHERE SUPERF>200 AND DEP<96;
SELECT LIBGEO,DEP
	FROM Communes
	WHERE MEDRFUC10>40000;
SELECT LIBGEO,DEP
	FROM Communes
	WHERE DECE11>NAIS11 AND P09POP>30000;
SELECT MAX(P09LOGVAC)
	FROM Communes ;
SELECT CODGEO AS CODGEOR, LIBGEO AS LIBGEOR, TCHOMB3T12
	FROM Rg;
SELECT LIBGEO AS Nom, LIBGEOD AS Departement
	FROM
		Communes JOIN Deptsok ON DEP=CODGEOD
	WHERE P09POPFaire >200000;
SELECT SUPERF AS Superficie, LIBGEO AS Nom, LIBGEOD AS Departement, LIBGEOR AS Region
	FROM
		Communes JOIN Deptsok JOIN Rgok ON DEP=CODGEOD AND REG=CODGEOR
	WHERE DEP<96 AND SUPERF>200;
SELECT SUPERF AS Superficie, LIBGEO AS Nom, LIBGEOD AS Departement, LIBGEOR AS Region
	FROM
		Communes JOIN Deptsok JOIN Rgok ON DEP=CODGEOD AND REG=CODGEOR WHERE DEP<96 AND
SUPERF>200
	GROUP BY SUPERF;
SELECT P09POP AS Population, LIBGEO AS Nom, LIBGEOD AS Departement, LIBGEOR AS Region
	FROM
		Communes JOIN Deptsok JOIN Rgok ON DEP=CODGEOD AND REG=CODGEOR
	WHERE DEP<96 AND Population>100000
	GROUP BY (- Population);
```