Contents

1 test

toto	tata
1	2
3.14	6.28

```
print(tabtest)
```

1.1 réglages

Drug	Patients
X	232
Y	3?1
Z	123

```
print(tbl)
```

2 TP

2.1 saisie du tableau

```
print(tbl[:][1])
```

[5, -18]

z(km)	T(°C)
0	15
5	-18
10	-49
12	-56
20	-56
-51	-46
30	-37
35	-22
40	-8
45	-2
48	-2
52	-7
55	-17
60	-33
65	-54
70	-65
75	-79
80	-86
84	-86
92	-86
95	-81
100	-72

2.1.1 parenthese pandas

```
import pandas as pd
D = pd.DataFrame(tbl).iloc[:, :]
print(D)
```

0 1 0 0 15 1 5 -18 2 10 -49 3 12 -56 4 20 -56 5 -51 -46 6 30 -37 7 35 -22 8 40 -8 9 45 -2 10 48 -2 11 52 -7 12 55 -17 13 60 -33 14 65 -54 15 70 -65 16 75 -79 17 80 -86 18 84 -86 19 92 -86 20 95 -81 21 100 -72

```
import pandas as pd
D = pd.DataFrame(tbl).iloc[:, 1:3]
print(D)
```

1 0 15 1 -18 2 -49 3 -56 4 -56 5 -46 6 -37 7 -22 8 -8 9 -2 10 -2 11 -7 12 -17 13 -33 14 -54 15 -65 16 -79 17 -86 18 -86 19 -86 20 -81 21 -72

```
#Altitude
print(D)
```

1 0 15 1 -18 2 -49 3 -56 4 -56 5 -46 6 -37 7 -22 8 -8 9 -2 10 -2 11 -7 12 -17 13 -33 14 -54 15 -65 16 -79 17 -86 18 -86 19 -86 20 -81 21 -72

2.2 values

```
M = 29.0e-3
R = 8.31
P0 = 1.0e5
g0 = 9.8
RT = 6.4e3
```

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

```
zexp = np.array([0.0, 5.0, 10.0, 12.0, 20.0, 25.0, 30.0, 35.0, 40.0, 45.0, 48.0, 52.0, 55.0, 60.0, 65.0, 70.0, 75.0, 80.0, 84.0,

Texp = np.array([15.0, -18.0, -49.0, -56.0, -56.0, -51.0, -46.0, -37.0, -22.0, -8.0, -2.0, -2.0, -7.0, -17.0, -33.0, -54.0, -65.

# print(len(zexp))
# print(len(Texp))
# print(zexp)
# print(Texp)
```

2.3 interpolation

On considère deux points de mesure i et $i+1$, on a la relation

$$T_k = az_k + b$$

avec a et b indéterminés. Evcrivons la relation de la température en $k = i$ et $k = i + 1$

```
def T(z,unite):
    z_km = z / 1000 #conversion
    alpha = 1 # facteur de conversion

    if unite == 'C':
        alpha = 0

    i = 0
    while z_km > zexp[i+1]: # recherche de l'indice i
        i = i + 1

    rate =  ( Texp[i+1] - Texp[i] ) / ( zexp[i+1] - zexp[i] )
    temperature = alpha*273 + Texp[i] + rate * (z_km - zexp[i])
    return temperature
```

2.4 gravity

```
def g(z):
    return g0 * RT**2 / (RT + z) **2
```

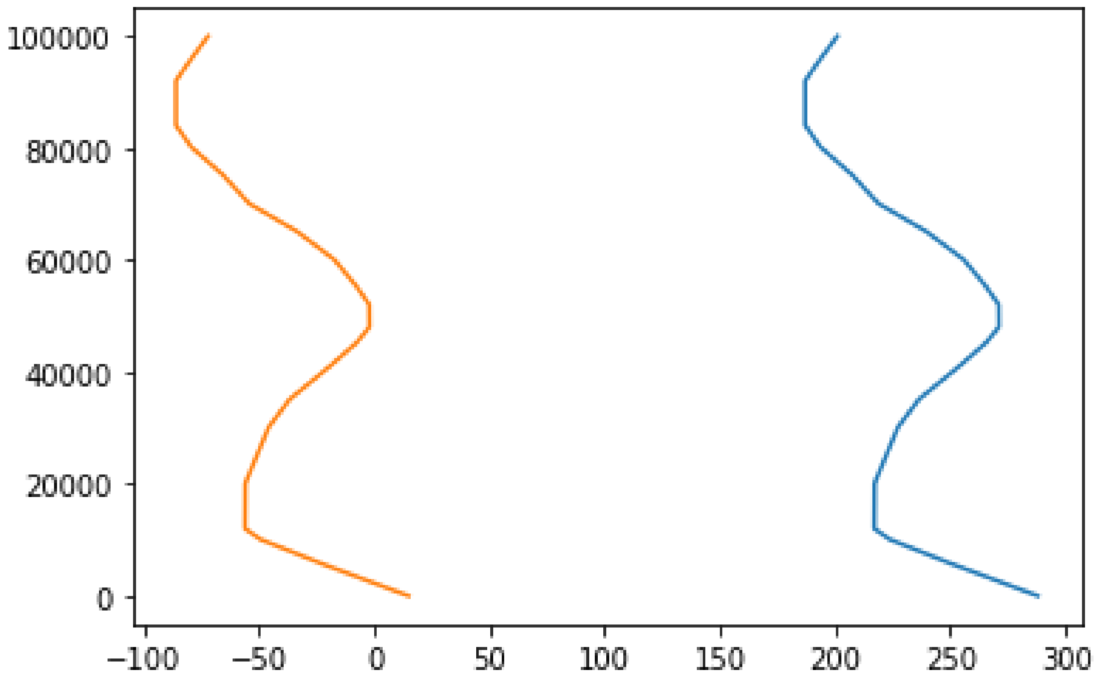
2.5 temperature

```
N = 1000
zmax = 100.0e3
dz = zmax / (N-1)
print(N, zmax, dz)
zatm = np.array([ k * dz for k in range(N) ])
```

```
Tatm = np.array([ T(zatm[k], 'C') for k in range(N) ])
TatmK = np.array([ T(zatm[k], 'K') for k in range(N) ])
gatm = np.array([ g(zatm[k]) for k in range(N)])
```

1000 100000.0 100.10010010010011

```
fig, ax = plt.subplots()
ax.plot( TatmK,zatm)
ax.plot( Tatm,zatm)
plt.savefig("ffffffffff")
```



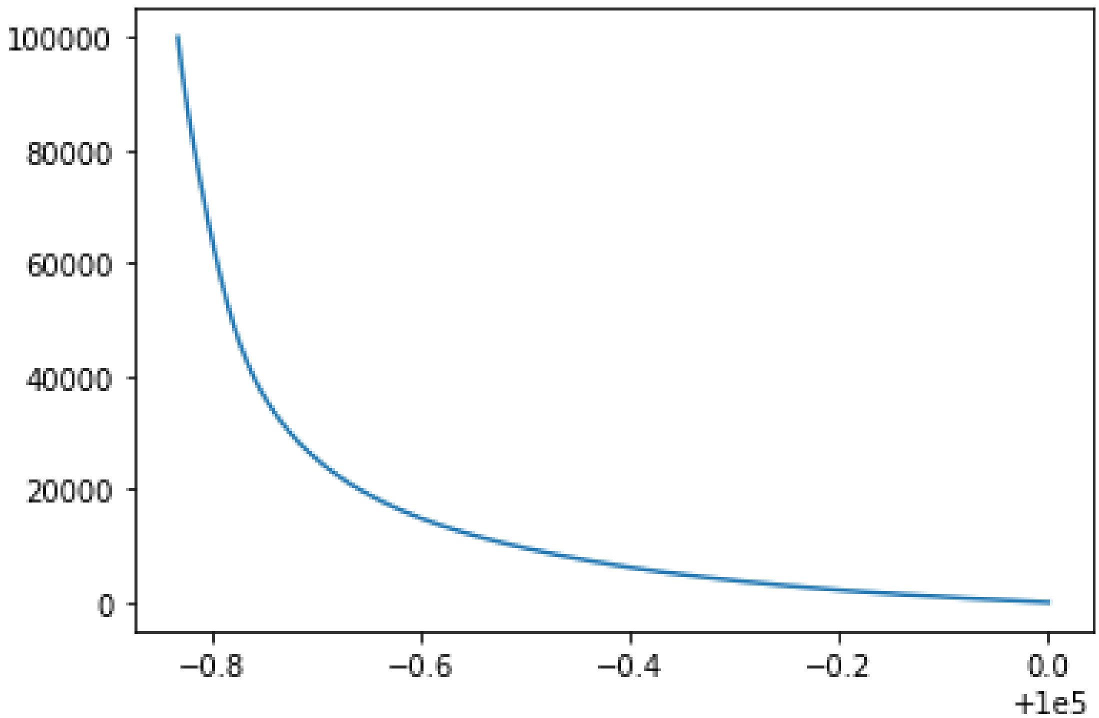
2.6 pressure

calcul du champ de pression par la méthode d'Euler

```
Patm = [P0]
gatm = [g0]
deltap = 0
gradient = 0
for k in range(N-1):
    gradient = - M * g( zatm[k] ) / (R * TatmK[k] )
    deltap = gradient * dz
    Patm.append( Patm[k] + deltap )
#    gatm.append( gatm[k] )
Patm = np.array(Patm)
print(M,R,P0,g0,RT)
```

0.029 8.31 100000.0 9.8 6400.0

```
plt.plot(Patm,zatm)
```



2.7 masse d'air

calcul de la masse d'air par la méthode des rectangles situé entre deux sphères d'altitude z et z+dz

```
def masse_atm(z):
    masse = 0
    k = 0
    while zatm[k] < z:
        dm = dz * 4 * np.pi * (RT + z)**2 * M * Patm[k] / (R* T(zatm[k],'K'))
        masse = masse + dm
        k = k+1
    return masse
```

```
mtot = masse_atm(100e3)
print(mtot)
```

2.201395425007424e+16

2.8 next