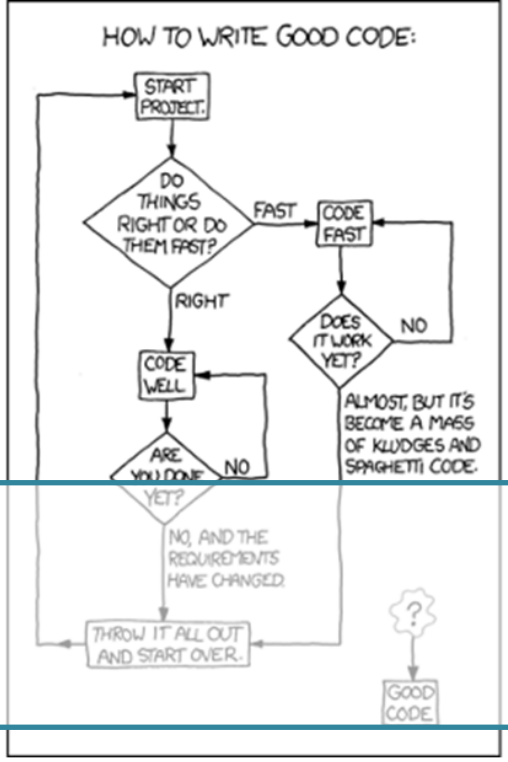
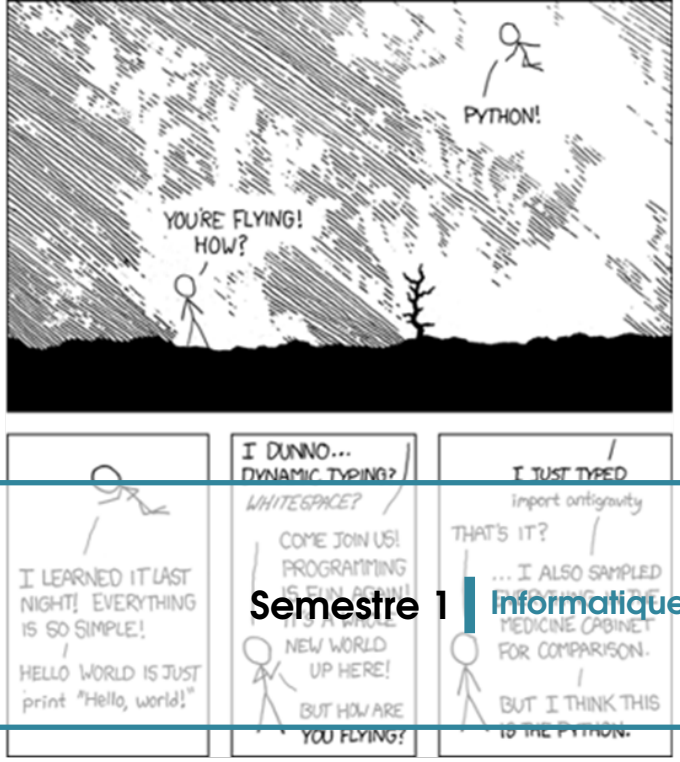




http://xkcd.com/353/
http://xkcd.com/844/

Thèmes d'étude

1 Bases de Python

Exercice 1 –

Question 1 *Évaluer les expressions suivantes en repérant auparavant celles qui donnent des résultats de type* `int`.

a) `4+2`
b) `25-3`
c) `-5+1`
d) `117*0`
e) `6*7-1`
f) `52*(3-5)`
g) `5*(-2)`
h) `22/(16-2*8)`
i) `42/6`
j) `18/7`
k) `(447+3*6)/5`
l) `0/0`

Question 2 *Calculer les restes et les quotients des divisions euclidiennes suivantes :*

a) $127 \div 8$
b) $54 \div 3$
c) $58 \div 5$
d) $58 \div (-5)$
e) $-58 \div 5$
f) $-58 \div (-5)$
g) $17583 \div 10$
h) $17583 \div 100$
i) $17583 \div 10^4$
j) $(2^7+2^4+2) \div 2^5$
k) $(2^7+2^4+2) \div 2^7$
l) $(2^7+2^4+2) \div 2^{10}$

Question 3 *Calculer les nombres suivants avec une expression Python en repérant auparavant ceux qui donnent un résultat de type* `int`.

a) 3^5
b) 2^{10}
c) $(-3)^7$
d) -3^7
e) 5^{-2}
f) 7^{5^4}
g) 7^{5^4}
h) 5^{7+6}
i) 5^7+6
j) 2^{10^4}

Question 4 *Évaluer les expressions suivantes.*

a) `4.3+2`
b) `2.5-7.3`
c) `42+4.`
d) `42+4`
e) `42.+4`
f) `12*0.`
g) `11.7*0`
h) `2,22/(1.6-2*0.8)`
i) `42/6`
j) `1,8/7`
k) `(447+3*6)/5`
l) `0/0`

Question 5 *Calculer, sans utiliser la fonction* `sqrt` *ni la division flottante* `/`*, les nombres suivants.*

a) $\dfrac{1}{7,9}$
b) $\sqrt{6,2}$
c) $\dfrac{1}{\sqrt{3,5}}$
d) $2\sqrt{2}$

De base, on ne peut réaliser que des calculs élémentaires avec Python. Cependant, il est possible d'avoir accès à des possibilités de calcul plus avancées en utilisant une *bibliothèque*. Par exemple, la bibliothèque `math` permet d'avoir accès à de nombreux outils mathématiques. On peut donc saisir

```
from math import sin, cos, tan, pi, e
from math import sin, cos, tan, pi, e
```

pour avoir accès à toutes ces fonctions.

Question 6 *Calculer les nombres suivants (on n'hésitera pas à consulter l'aide en ligne).*

a) e^2
b) $\sqrt{13}$
c) $\cos\left(\dfrac{\pi}{5}\right)$
d) $e^{\sqrt{5}}$
e) $\ln 2$
f) $\ln 10$
g) $\log_2 10$
h) $\tan\left(\dfrac{\pi}{2}\right)$

Question 7 *Les expressions suivantes sont-elles équivalentes?*

a) `8.5 / 2.1`
b) `int(8.5) /`
`int(2.1)`
c) `int(8.5 / 2.1)`

Et celles-ci?

a) `float(8 * 2)`
b) `8 * 2`
c) `8. * 2.`

Prévoir la valeur des expressions suivantes puis vérifier cela (avec IDLE).

a) `1.7 + 1.3`
b) `2 - 1`
c) `2. - 1`
d) `2 - 1.`
e) `(2 - 1).`
f) `.5 + .5`
g) `4 / (9 - 3**2)`
h) `4 / (9. - 3**2)`

Question 8 *Déterminer de tête la valeur des expressions suivantes avant de le vérifier (avec IDLE).*

a) `0 == 42`
b) `1 = 1`
c) `3 == 3.`
d) `0 != 1`
e) `0 < 1`
f) `4. >= 4`
g) `0 !< 1`
h) `2*True + False`
i) `-1 <= True`
j) `1 == True`
k) `False != 0.`
l) `True and False`
m) `True or False`
n) `True or True`
o) `(2 == 3-1) or (1/0 == 5)`
p) `(1/0 == 5) or (2 == 3-1)`
q) `True or True and False`
r) `False or True and False`
s) `not (1 == 1 or 4 == 5)`
t) `(not 1 == 1) or 4 == 5`
u) `not True or True`

Question 9 *Dans votre IDE, cliquer sur File/New File. Une nouvelle fenêtre apparaît. Dans cette fenêtre, taper les lignes suivantes.*

```
3*2
print(2*3.)
17*1.27
```

Enregistrer le document produit puis, toujours dans cette fenêtre, exécutez-le. Observez le résultat dans l'interpréteur interactif. Modifiez les instructions pour que tous les résultats de calcul s'affichent dans l'interpréteur interactif.

Exercice 2 –

Question 1 *Calculer les restes et les quotients des divisions euclidiennes suivantes*

a) $127 \div 8$
b) $54 \div 3$
c) $58 \div 5$
d) $58 \div (-5)$
e) $-58 \div 5$
f) $-58 \div (-5)$
g) $17583 \div 10$
h) $17583 \div 100$
i) $17583 \div 10^4$
j) $(2^7+2^4+2) \div 2^5$
k) $(2^7+2^4+2) \div 2^7$
l) $(2^7+2^4+2) \div 2^{10}$

Question 2 *Calculer les nombres suivants avec une expression Python en repérant auparavant ceux qui donnent un résultat de type* `int`.

a) 3^5
b) 2^{10}
c) $(-3)^7$
d) -3^7
e) 5^{-2}
f) $7^{(5^4)}$
g) $(7^5)^4$
h) 5^{7+6}
i) 5^7+6
j) $2^{(10^4)}$

Exercice 3 –

Question 1 *Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif d'IDLE.*

a) `[1,2,3,"a"]`
b) `123a`
c) `[]`
d) `[]+[]`
e) `[]+[] == []`
f) `[1,2] + [5,7,9]`
g) `[0,0]+[0]`
h) `len(["a","b"])`
i) `len([])`
j) `len([[]])`
k) `len([[[]]])`
l) `len([0,0]+[1])`

Question 2 *Calculer cette suite d'expressions.*

```
[i for i in range(10)]
[compt**2 for compt in range(7)]
[j+1 for j in range(-2,8)]
```

Sur ce modèle, obtenir de manière synthétique :

a) la liste des 20 premiers entiers naturels impairs ;
b) la liste de tous les multiples de 5 entre 100 et 200 (inclus) ;
c) La liste de tous les cubes d'entiers naturels, inférieurs ou égaux à 1000.
d) une liste contenant tous les termes entre -20 et 5 d'une progression arithmétique de raison 0,3 partant de -20.

Exercice 4 –

Question 1 *Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif.*

a) `(1,2)`
b) `(1)`
c) `(1,)`
d) `(,)`
e) `()`
f) `()+()`
g) `()+() == ()`
h) `(1,2)+3`
i) `(1,2)+(3)`
j) `(1,2)+(3,)`
k) `(1,2)+(3,4,5)`
l) `len((1,7,2,"zzz",[]))`
m) `len(())`
n) `len(("a","bc")+("cde",""))`

Question 2 *Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif.*

a)
```
t = (2,"abra",9,6*9,22)
print(t)
t[0]
t[-1]
t[1]
t[1] = "cadabra"
```

b)
```
res = (45,5)
x,y = res
(x,y) == x,y
(x,y) == (x,y)
print x
print(y)
x,y = y,x
print(y)
```

c)
```
v = 7
ex = (-1,5,2,"","abra",8,3,v)
5 in ex
abra in ex
(2 in ex) and ("abr" in ex)
v in ex
```

Question 3 *Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif.*

a) `"abba"`
b) `abba`
c) `""`
d) `"" == " "`
e) `""+""`
f) `""+"" == ""`
g) `"May"+" "+"04th"`
h) `"12"+3`
i) `"12"+"trois"`
j) `len("abracadabra")`
k) `len("")`
l) `len("lamartin"+"2015")`

Question 4 *Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif.*

a)
```
t = "oh␣oui␣youpi␣!"
print(t)
t[0]
t[-1]
t[1]
t[2]
t[1] = "o"
```

b)
```
ex = "abdefgh"
"a" in ex
a in ex
"def" in ex
"adf" in ex
```

Question 5 *Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif d'IDLE.*

a) `[1,2,3,"a"]`
b) `123a`
c) `[]`
d) `[]+[]`
e) `[]+[] == []`
f) `[1,2] + [5,7,9]`
g) `[0,0]+[0]`
h) `len(["a","b"])`
i) `len([])`
j) `len([[]])`
k) `len([[[]]])`
l) `len([0,0]+[1])`

Question 6 *Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif d'IDLE.*

a)
```
t = [1,2,3,4,5,6]
u = ["a","b","c","d"]
print(t+u)
t[0]
t[-1]
z = t[3]
print(z)
t.append(7)
print(t)
```

c)
```
ex = ["sin","cos","tan","log","exp"]
"log" in ex
log in ex
"l" in ex
z = ex.pop()
print(z)
z in ex
print(ex)
```

d)
```
u = [1,2,3,4,5,6]
L = u
u = [1,2,3,42,5,6]
print(L)
```

e)
```
u = [1,2,3,4,5,6]
L = u
u[3] = 42
print(L)
```

Question 7 *Calculer cette suite d'expressions.*

```
[i for i in range(10)]
[compt**2 for compt in range(7)]
[j+1 for j in range(-2,8)]
```

Sur ce modèle, obtenir de manière synthétique :
a) la liste des 20 premiers entiers naturels impairs ;
b) la liste de tous les multiples de 5 entre 100 et 200 (inclus) ;
c) La liste de tous les cubes d'entiers naturels, inférieurs ou égaux à 1000.
d) une liste contenant tous les termes entre -20 et 5 d'une progression arithmétique de raison 0,3 partant de -20.

Question 8

a) Affecter à `v` la liste `[2,5,3,-1,7,2,1]`
b) Affecter à `L` la liste vide.
c) Vérifier le type des variables crées.
d) Calculer la longueur de `v`, affectée à `n` et celle de `L`, affectée à `m`.
e) Tester les expressions suivantes : `v[0]`, `v[2]`, `v[n]`, `v[n-1]`, `v[-1]` et `v[-2]`.
f) Changer la valeur du quatrième élément de `v`.
g) Que renvoie `v[1:3]` ? Remplacer dans `v` les trois derniers éléments par leurs carrés.
h) Que fait `v[1] = [0,0,0]` ? Combien d'éléments y a-t-il alors dans `v` ?

Question 9 *Quel type choisiriez-vous pour représenter les données suivantes ?*
Vous justifierez brièvement chaque réponse.
a) Le nom d'une personne.
b) L'état civil d'une personne : nom, prénom, date de naissance, nationalité.
c) Les coordonnées d'un point dans l'espace.
d) L'historique du nombre de 5/2 dans la classe de MP du lycée.
e) Un numéro de téléphone.
f) *Plus difficile :* l'arbre généalogique de vos ancêtres.

Question 10
Evaluer les expressions suivantes.

a) `4.3+2`
b) `2.5-7.3`
c) `42+4.`
d) `42+4`
e) `42.+4`
f) `12*0.`
g) `11.7*0`
h) `2,22/(1.6-2*0.8)`
i) `42/6`
j) `1,8/7`
k) `(447+3*6)/5`
l) `0/0`

Exercice 5 –

Question 1 *Voici des affectations successives des variables a et b. Dresser un tableau donnant les valeurs de a et b à chaque étape.*

```
>>> a = 1
>>> b = 5
>>> a = b-3
>>> b = 2*a
>>> a = a
>>> a = b
```

Question 2 *Écrire une séquence d'instructions qui échange les valeurs de deux variables x et y.*

Question 3 *Écrire, sans variable supplémentaire, une suite d'affectation qui permute circulairement vers la gauche les valeurs des variables x, y, z : x prend la valeur de y qui prend celle de z qui prend celle de x.*

Question 4 *Calculer, sans utiliser la fonction* `sqrt` *ni la division flottante* `/`*, les nombres suivants.*

a) $\dfrac{1}{7,9}$
b) $\sqrt{6,2}$
c) $\dfrac{1}{\sqrt{3,5}}$
d) $2\sqrt{2}$

Modéliser

De base, on ne peut réaliser que des calculs élémentaires avec Python. Cependant, il est possible d'avoir accès à des possibilités de calcul plus avancées en utilisant une *bibliothèque*. Par exemple, la bibliothèque `math` permet d'avoir accès à de nombreux outils mathématiques. On peut donc taper

```
from math import sqrt, log, exp, sin, cos, tan
    , pi, e
```

pour avoir accès à toutes ces fonctions.
Calculer les nombres suivants (on n'hésitera pas à consulter l'aide en ligne).

a) e^2
b) $\sqrt{13}$
c) $\cos\left(\frac{\pi}{5}\right)$
d) $e^{\sqrt{5}}$
e) $\ln 2$
f) $\ln 10$
g) $\log_2 10$
h) $\tan\left(\frac{\pi}{2}\right)$

Exercice 6 –

Question 1 *Voici des affectations successives des variables a et b. Dresser un tableau donnant les valeurs de a et b à chaque étape.*

```
>>> a = 1
>>> b = 5
>>> a = b-3
>>> b = 2*a
>>> a = a
>>> a = b
```

Question 2 *Écrire une séquence d'instructions qui échange les valeurs de deux variables x et y.*

Question 3 *Écrire, sans variable supplémentaire, une suite d'affectation qui permute circulairement vers la gauche les valeurs des variables x, y, z : x prend la valeur de y qui prend celle de z qui prend celle de x.*

Question 4 *Dans les cas où c'est possible, affecter les valeurs aux variables correspondantes à l'aide de l'interpréteur interactif. On notera* `var ← a` *pour dire que l'on affecte la valeur* `a` *à la variable* `var`.

a) `ArthurDent ← 42`
b) `4 ← 0.`
c) `L ← []`
d) `list ← [1,2,3]`
e) `int ← 5`
f) `s ← ""`
g) `True ← 1`
h) `ok ← ok`
i) `x ← "x"`
j) `a ← 1<0`
k) `lam ← 1/0`
l) `or ← "xor"`

Question 5 *On part des affectations suivantes :* `a ← 5` *et* `b ← 0`. *Pour la suite d'instructions suivante, prévoir ligne à ligne le résultat affiché par l'interpréteur interactif de Python ainsi que l'état des variables. Le vérifier grâce à l'interpréteur interactif d'IDLE, en prenant soin de partir d'une nouvelle session.*

```
a*b
x = a**b + a
print(x)
print(y)
z = x
x = 5
print(z)
a = a+a**b
print(a)
```

Question 6 *Affecter des valeurs toutes différentes aux variables* `a`, `b`, `c` *et* `d`.

Á chaque fois, effectuer les permutations suivantes de manière naïve (c'est-à-dire, sans utiliser de `tuple`).

a) Échanger les contenus de `a` et de `b`.
b) Placer le contenu de `b` dans `a`, celui de `a` dans `c` et celui de `c` dans `b`.
c) Placer le contenu de `a` dans `d`, celui de `d` dans `c`, celui de `c` dans `b` et celui de `b` dans `a`.

Reprendre cet exercice en effectuant chaque permutation en une instruction à l'aide d'un `tuple`.

Question 7 *Combien d'affectations sont suffisantes pour permuter circulairement les valeurs des variables $x_1, \cdots, x_n$ sans utiliser de variable supplémentaire ? Et en utilisant autant de variables supplémentaires que l'on veut ?*

Question 8 *Mêmes questions en remplaçant suffisantes par nécessaires.*

Question 9 *Supposons que la variable x est déjà affectée, et soit $n \in \mathbb{N}$. On veut calculer x^n sans utiliser la puissance, avec uniquement des affectations, autant de variables que l'on veut, mais avec le moins de multiplications possible. Par exemple, avec les 4 instructions :*

```
>>> y1 = x * x
>>> y2 = y1 * x
>>> y3 = y2 * x
>>> y4 = y3 * x
```

on calcule x^5, qui est la valeur de `y4`.
Mais 3 instructions suffisent :

```
>>> y1 = x * x
>>> y2 = y1 * y1
>>> y3 = y2 * x
```

Question 10 *En fonction de n, et avec les contraintes précédentes, quel est le nombre minimum d'instructions pour calculer x^n ?*

Question 11 *Calculer, sans utiliser la fonction* `sqrt` *ni la division flottante* `/`, *les nombres suivants.*

a) $\frac{1}{7,9}$
b) $\sqrt{6,2}$
c) $\frac{1}{\sqrt{3,5}}$
d) $2\sqrt{2}$

De base, on ne peut réaliser que des calculs élémentaires avec Python. Cependant, il est possible d'avoir accès à des possibilités de calcul plus avancées en utilisant une

bibliothèque. Par exemple, la bibliothèque `math` permet d'avoir accès à de nombreux outils mathématiques. On peut donc taper

```
from math import sqrt, log, exp, sin, cos, tan
    , pi, e
```

pour avoir accès à toutes ces fonctions.
Calculer les nombres suivants (on n'hésitera pas à consulter l'aide en ligne).

a) e^2
b) $\sqrt{13}$
c) $\cos\left(\frac{\pi}{5}\right)$
d) $e^{\sqrt{5}}$
e) $\ln 2$
f) $\ln 10$
g) $\log_2 10$
h) $\tan\left(\frac{\pi}{2}\right)$

Exercice 7 –

Question 1

1. Ecrire une fonction qui à un nombre entier associe le chiffre des unités.
2. Ecrire une fonction qui à un nombre entier associe le chiffre des dizaines.
3. Ecrire une fonction qui à un nombre entier associe le chiffre des unités en base 8.

Question 2 *Ouvrir votre IDE, écrire la fonction suivante dans un fichier, l'enregistrer, taper* run *(F5) puis utiliser la fonction dans l'interpréteur interactif. Décrire ensuite précisément ce que réalise cette fonction.*

```
def split_modulo(n):
  """A vous de dire ce que fait cette fonction !
     """
  return (n%2,n%3,n%5)
```

Question 3 *Écrire une fonction* `norme` *qui prend en argument un vecteur de* $\mathbb{R}^2$ *donnée par ses coordonnées et renvoie sa norme euclidienne. Vous devrez spécifier clairement le type de l'argument à l'utilisateur via la* docstring.

Question 4 *Écrire une fonction* `lettre` *qui prend en argument un entier* `i` *et renvoie la* `i`^{e} *lettre de l'alphabet.*

Question 5 *Écrire une fonction* `carres` *qui prend en argument un entier naturel* `n` *et qui renvoie la liste des* `n` *premiers carrés d'entiers, en commençant par* 0.

Exercice 8 –

Question 1 *Déterminer de tête la valeur des expressions suivantes avant de le vérifier (avec IDLE).*

a) `0 == 42`
b) `1 = 1`
c) `3 == 3.`
d) `0 != 1`
e) `0 < 1`
f) `4. >= 4`
g) `0 !< 1`
h) `2*True + False`
i) `-1 <= True`
j) `1 == True`
k) `False != 0.`
l) `True and False`
m) `True or False`
n) `True or True`
o) `(2 == 3-1) or (1/0 == 5)`
p) `(1/0 == 5) or (2 == 3-1)`
q) `True or True and False`
r) `False or True and False`
s) `not (1 == 1 or 4 == 5)`
t) `(not 1 == 1) or 4 == 5`
u) `not True or True`

Exercice 9 –

Question 1 *Dans chaque cas, indiquez le type que vous utiliseriez pour modéliser les grandeurs suivantes dans leur contexte scientifique usuel. Vous justifierez brièvement chaque réponse.*

a) La taille d'un individu en mètres.
b) Le tour de taille d'un manequin, en millimètres.
c) Le nombre d'Avogadro.
d) Le nombre de Joules dans une calorie.
e) Le nombre de secondes dans une année.
f) Le plus grand nombre premier représentable avec 20 chiffres en écriture binaire.

Exercice 10 –

Question 1 *Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif.*

a) `(1,2)`
b) `(1)`
c) `(1,)`
d) `(,)`
e) `()`
f) `()+()`
g) `()+() == ()`
h) `(1,2)+3`
i) `(1,2)+(3)`
j) `(1,2)+(3,)`
k) `(1,2)+(3,4,5)`
l) `len((1,7,2,"zzz",[]))`
m) `len(())`
n) `len(("a","bc")+("cde",""))`

Exercice 11 –

Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif d'IDLE.

a)
```
t = (2,"abra",9,6*9,22)
print(t)
t[0]
t[-1]
t[1]
t[1] = "cadabra"
```

b)
```
res = (45,5)
x,y = res
(x,y) == x,y
(x,y) == (x,y)
print x
print(y)
x,y = y,x
print(y)
```

c)
```
v = 7
ex = (-1,5,2,"","abra",8,3,v)
5 in ex
```

Modéliser

```
abra in ex
(2 in ex) and ("abr" in ex)
v in ex
```

Exercice 12 –

Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif d'IDLE.

a) `"abba"`
b) `abba`
c) `""`
d) `"" == " "`
e) `""+""`
f) `""+"" == ""`
g) `"May"+" "+"04th"`
h) `"12"+3`
i) `"12"+"trois"`
j) `len("abracadabra")`
k) `len("")`
l) `len("lamartin"+"2015")`

Exercice 13 –

Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif d'IDLE.

a)
```
t = "oh␣oui␣youpi␣!"
print(t)
t[0]
t[-1]
t[1]
t[2]
t[1] = "o"
```

b)
```
ex = "abdefgh"
"a" in ex
a in ex
"def" in ex
"adf" in ex
```

Exercice 14 –

Prévoir les résultats des expressions suivantes, puis le vérifier grâce à l'interpréteur interactif d'IDLE.

a) `[1,2,3,"a"]`
b) `123a`
c) `[]`
d) `[]+[]`
e) `[]+[] == []`
f) `[1,2] + [5,7,9]`
g) `[0,0]+[0]`
h) `len(["a","b"])`
i) `len([])`
j) `len([[]])`
k) `len([[[]]])`
l) `len([0,0]+[1])`

Exercice 15 –

Question 1 *Pour chaque séquence d'instruction, prévoir son résultat puis le vérifier grâce à l'interpréteur interactif d'IDLE.*

a)
```
t = [1,2,3,4,5,6]
u = ["a","b","c","d"]
print(t+u)
t[0]
t[-1]
z = t[3]
print(z)
t.append(7)
print(t)
```

b)
```
ex = ["sin","cos","tan","log","exp"]
"log" in ex
log in ex
"l" in ex
z = ex.pop()
print(z)
z in ex
print(ex)
```

c)
```
u = [1,2,3,4,5,6]
L = u
u = [1,2,3,42,5,6]
print(L)
```

d)
```
u = [1,2,3,4,5,6]
L = u
u[3] = 42
print(L)
```

Exercice 16 –

Question 1 *Calculer cette suite d'expressions.*

```
[i for i in range(10)]
[compt**2 for compt in range(7)]
[j+1 for j in range(-2,8)]
```

Sur ce modèle, obtenir de manière synthétique :

a) la liste des 20 premiers entiers naturels impairs ;
b) la liste de tous les multiples de 5 entre 100 et 200 (inclus) ;
c) La liste de tous les cubes d'entiers naturels, inférieurs ou égaux à 1000.
d) une liste contenant tous les termes entre -20 et 5 d'une progression arithmétique de raison 0,3 partant de -20.

Exercice 17 –

Question 1

1. Affecter à `v` la liste `[2,5,3,-1,7,2,1]`
2. Affecter à `L` la liste vide.
3. Vérifier le type des variables crées.
4. Calculer la longueur de `v`, affectée à `n` et celle de `L`, affectée à `m`.
5. Tester les expressions suivantes : `v[0]`, `v[2]`, `v[n]`, `v[n-1]`, `v[-1]` et `v[-2]`.
6. Changer la valeur du quatrième élément de `v`.
7. Que renvoie `v[1:3]` ? Remplacer dans `v` les trois derniers éléments par leurs carrés.
8. Que fait `v[1] = [0,0,0]` ? Combien d'éléments y a-t-il alors dans `v` ?

Exercice 18 –

Question 1 *Quel type choisiriez-vous pour représenter les données suivantes ?*
Vous justifierez brièvement chaque réponse.

1. Le nom d'une personne.
2. L'état civil d'une personne : nom, prénom, date de naissance, nationalité.
3. Les coordonnées d'un point dans l'espace.
4. L'historique du nombre de 5/2 dans la classe de MP du lycée.

5. Un numéro de téléphone.
6. *Plus difficile :* l'arbre généalogique de vos ancêtres.

Exercice 19 –

Question 1 *Quel type choisiriez-vous pour représenter les données suivantes ?*
Vous justifierez brièvement chaque réponse.

a) Le nom d'une personne.
b) L'état civil d'une personne : nom, prénom, date de naissance, nationalité.
c) Les coordonnées d'un point dans l'espace.
d) L'historique du nombre de 5/2 dans la classe de MP du lycée.
e) Un numéro de téléphone.
f) *Plus difficile :* l'arbre généalogique de vos ancêtres.

Exercice 20 –

Question 1 *Dans les cas où c'est possible, affecter les valeurs aux variables correspondantes à l'aide de l'interpréteur interactif. On notera* `var ← a` *pour dire que l'on affecte la valeur* `a` *à la variable* `var`.

a) `ArthurDent ← 42`
b) `4 ← 0.`
c) `L ← []`
d) `list ← [1,2,3]`
e) `int ← 5`
f) `s ← ""`
g) `True ← 1`
h) `ok ← ok`
i) `x ← "x"`
j) `a ← 1<0`
k) `lam ← 1/0`
l) `or ← "xor"`

Exercice 21 –

Question 1 *On considère les affectations* `a ← -1.` *et* `b ← 5`. *Prévoir la valeur de chacune de ces expressions, puis le vérifier à l'aide de l'interpréteur interactif.*

a) `a * a`
b) `a ** a`
c) `a ** a == a`
d) `a * b`
e) `a+b == 5`
f) `a+6>=b`
g) `b<100 | a**2==-1`
h) `b-3`

Exercice 22 –

Question 1 *On part des affectations suivantes :* `a ← 5` *et* `b ← 0`. *Pour la suite d'instructions suivante, prévoir ligne à ligne le résultat affiché par l'interpréteur interactif de Python ainsi que l'état des variables. Le vérifier grâce à l'interpréteur interactif d'IDLE, en prenant soin de partir d'une nouvelle session.*

```
a*b
x = a**b + a
print(x)
print(y)
z = x
x = 5
print(z)
a = a+a**b
print(a)
```

Exercice 23 –

Question 1 *Affecter des valeurs toutes différentes aux variables* `a`, `b`, `c` *et* `d`.

À chaque fois, effectuer les permutations suivantes de manière naïve (c'est-à-dire, sans utiliser de `tuple`).

a) Échanger les contenus de `a` et de `b`.
b) Placer le contenu de `b` dans `a`, celui de `a` dans `c` et celui de `c` dans `b`.
c) Placer le contenu de `a` dans `d`, celui de `d` dans `c`, celui de `c` dans `b` et celui de `b` dans `a`.

Reprendre cet exercice en effectuant chaque permutation en une instruction à l'aide d'un `tuple`.

Exercice 24 –

Question 1

a) Affecter à la variable `mon_age` l'âge que vous aviez il y a 13 ans.
b) Écrire l'opération qui vous permet d'actualiser votre âge, tout en conservant la même variable.
c) Que donne l'interpréteur après exécution des expressions suivantes ? Pourquoi ?

```
mon-age = 18
2013_mon_age = 18
True = 18
```

d) À partir d'une nouvelle session d'IDLE, exécuter les expressions suivantes et commenter le résultat.

```
age = 5
age = Age + 14
```

Exercice 25 –

Question 1 *Combien d'affectations sont suffisantes pour permuter circulairement les valeurs des variables $x_1, \cdots, x_n$ sans utiliser de variable supplémentaire ? Et en utilisant autant de variables supplémentaires que l'on veut ?*

Question 2 *Mêmes questions en remplaçant suffisantes par nécessaires.*

Exercice 26 –

Supposons que la variable x est déjà affectée, et soit $n \in \mathbb{N}$. On veut calculer x^n sans utiliser la puissance, avec uniquement des affectations, autant de variables que l'on veut, mais avec le moins de multiplications possible. Par exemple, avec les 4 instructions :

```
>>> y1 = x * x
>>> y2 = y1 * x
>>> y3 = y2 * x
>>> y4 = y3 * x
```

on calcule x^5, qui est la valeur de `y4`.
Mais 3 instructions suffisent :

```
>>> y1 = x * x
>>> y2 = y1 * y1
>>> y3 = y2 * x
```

En fonction de n, et avec les contraintes précédentes, quel est le nombre minimum d'instructions pour calculer x^n ?

Exercice 27 –

Question 1 *Voici des affectations successives des variables a et b. Dresser un tableau donnant les valeurs de a et b à chaque étape.*

```
a = 1
b = 5
a = b-3
b = 2*a
a = a
a = b
```

Exercice 28 –

Question 1 *Écrire une séquence d'instructions qui échange les valeurs de deux variables x et y.*

Exercice 29 –

Question 1 *Écrire, sans variable supplémentaire, une suite d'affectation qui permute circulairement vers la gauche les valeurs des variables x, y, z : x prend la valeur de y qui prend celle de z qui prend celle de x.*

Exercice 30 –

Question 1 *Ouvrir votre IDE, écrire la fonction suivante dans un fichier, l'enregistrer, taper* run *(F5) puis utiliser la fonction dans l'interpréteur interactif. Décrire ensuite précisément ce que réalise cette fonction.*

```
def split_modulo(n):
  """A vous de dire ce que fait
      cette fonction !"""
  return (n%2,n%3,n%5)
```

Exercice 31 –

Question 1 ´
Ecrire une fonction `moy_extr(L)` qui prend en argument une liste `L` et renvoie en sortie la moyenne du premier et du dernier élément de `L`.

Exercice 32 –

Question 1 *Écrire une fonction* `norme` *qui prend en argument un vecteur de $\mathbb{R}^2$ donnée par ses coordonnées et renvoie sa norme euclidienne. Vous devrez spécifier clairement le type de l'argument à l'utilisateur via la* docstring.

Exercice 33 –

Question 1 *Écrire une fonction* `lettre` *qui prend en argument un entier* `i` *et renvoie la* $\texttt{i}^e$ *lettre de l'alphabet.*

Exercice 34 –

Question 1 *Écrire une fonction* `carres` *qui prend en argument un entier naturel* `n` *et qui renvoie la liste des* `n` *premiers carrés d'entiers, en commençant par* 0.

Exercice 35 –

Question 1 *On cherche à écrire une fonction prenant en argument une liste d'entiers et incrémentant de* 1 *le premier élément de cette liste.*

a) Écrire une telle fonction `incr_sans_effet_de_bord`, qui ne modifie pas la liste initiale et renvoie en sortie une nouvelle liste.
b) Écrire une telle fonction `incr_avec_effet_de_bord`, qui modifie la liste initiale et ne renvoie rien en sortie (ponctuer par un `return None`).

Exercice 36 –

Question 1
Écrire une fonction `appartient(a,c,r)` prenant en argument
- un couple de nombres `a`;
- un couple de nombres `c`;
- un nombre positif `r`;

et renvoyant la valeur de vérité de « le point de coordonnées `a` est dans le disque fermé de centre de coordonnées `c` et de rayon `r` ».

Exercice 37 –

Question 1

1. Ecrire une fonction qui à un nombre entier associe le chiffre des unités.
2. Ecrire une fonction qui à un nombre entier associe le chiffre des dizaines.
3. Ecrire une fonction qui à un nombre entier associe le chiffre des unités en base 8.

Exercice 38 –

Question 1
Écrire une fonction `moy_extr(L)` qui prend en argument une liste `L` non vide et renvoie en sortie la moyenne du premier et du dernier élément de `L`.

Question 2
On cherche à écrire une fonction prenant en argument une liste d'entiers (non vide) et incrémentant de 1 le premier élément de cette liste.

a) Écrire une telle fonction `incr_sans_effet_de_bord`, qui ne modifie pas la liste initiale et renvoie en sortie une nouvelle liste.
b) Écrire une telle fonction `incr_avec_effet_de_bord`, qui modifie la liste initiale et ne renvoie rien en sortie (ponctuer par un `return None`).

Exercice 39 –

Question 1 *Indenter de deux manières différentes la suite d'instructions suivante afin que la variable* `t` *contienne* `True` *pour une indentation, puis* `False` *pour l'autre.*

```
x = 0
y = 5
t = False
if x>=1:
t = True
if y <= 6:
t = True
```

Exercice 40 –

Question 1 *Réécrire la suite d'instructions suivante de manière plus appropriée.*

```
from random import randrange
# Un entier aléatoire entre 0 et 99
n = randrange(100)
if n <= 10:
    print("Trop␣petit")
else:
    if n >= 50:
        print("Trop␣grand")
    else:
        print("Juste␣comme␣il␣faut")
```

Exercice 41 –

a) Écrire une fonction `neg(b)` qui renvoie la négation du booléen `b` sans utiliser `not`.

b) Écrire une fonction `ou(a,b)` qui renvoie le ou logique des booléen `a` et `b` sans utiliser `not`, `or` ni `and`.

c) Écrire une fonction `et(a,b)` qui renvoie le et logique des booléen `a` et `b` sans utiliser `not`, `or` ni `and`.

Exercice 42 –

Indenter de deux manières différentes la suite d'expressions suivante de manière à ce qu'à son exécution, le programme affiche soit la liste de tous les éléments de L inférieurs ou égaux à m, soit juste le dernier.

```
from random import randrange
L = [randrange(100) for i in range(100)] # 100
    valeurs entre 0 et 99.
m = 50
for x in L:
if x <= m:
p = x
print(p)
```

Exercice 43 –

Question 1 *Que fait la fonction suivante ? La corriger pour qu'elle coïncide avec le but annoncé.*

```
def inv(n):
    """Somme des inverses des n premiers
       entiers naturels non nuls"""
    s = 0
    for k in range(n):
        x = 1/k
    s = s+x
    return s
```

Exercice 44 –

Décrire ce que fait cette suite d'instructions.

```
from random import randrange
# Un entier entre 0 et 999
n = randrange(1000)
# n+1 valeurs entre 0 et 99.
L = [randrange(100) for i in range(n+1)]
p = 0
for x in L:
    if x <= 10:
        p = p + x**2
```

Et celle-ci ?

```
from random import randrange
# Un entier entre 0 et 999
n = randrange(1000)
# n+1 valeurs entre 0 et 99.
L = [randrange(100) for i in range(n+1)]
p = 0
for i in range(n+1):
    if L[i] <= 10:
        p = p + i**2
```

Exercice 45 –

Écrire une suite d'instructions permettant de calculer la somme des racines carrées des cinquante premiers entiers naturels non nuls.

Exercice 46 –

Écrire la suite d'instructions suivantes dans un fichier, l'enregistrer puis l'exécuter (F5). À l'instant qui vous convient, presser Ctrl+C.

```
a = 1
while a>0:
    a = a+1
```

Exercice 47 –

Question 1 *Que fait la fonction suivante ? La corriger pour qu'elle coïncide avec le but annoncé.*

```
def sqrt_int(n):
    """Renvoie la partie entière de la racine
        carrée de n"""
    s = 0
    while s**2 <= n:
        s = s+1
        s = s-1
    return s
```

Exercice 48 –

Question 1

Un taupin se lance dans un marathon d'exercices, mais se fatigue vite. Il réalise le i^e exercice en $\sqrt{i}$ minutes. Combien d'exercices arrive-t-il à faire en 4 heures ? Pour faciliter la correction, on écrira une fonction `nb_exos()` ne prenant pas d'argument et renvoyant le résultat demandé.

Exercice 49 –

Expliquer et justifier ce que fait la fonction suivante.

```
def cesar(k):
    """?????"""
    alphabet = "abcdefghijklmnopqrstuvxyz"
    s = ""
    for i in range(26):
        s = s + alphabet[i+k % 26]
    return s
```

Exercice 50 –

Question 1 *Indenter de deux manières différentes la suite d'instructions suivante afin que la variable* `t` *contienne* `True` *pour une indentation, puis* `False` *pour l'autre.*

```
x = 0
y = 5
t = False
if x>=1:
t = True
if y <= 6:
t = True
```

Question 2 *Réécrire la suite d'instructions suivante de manière plus appropriée.*

```
from random import randrange
# Un entier aléatoire entre 0 et 99
n = randrange(100)
if n <= 10:
    print("Trop␣petit")
else:
    if n >= 50:
        print("Trop␣grand")
    else:
        print("Juste␣comme␣il␣faut")
```

Question 3 *Que fait la fonction suivante ? La corriger pour qu'elle coïncide avec le but annoncé.*

```
def inv(n):
    """Somme des inverses des n premiers
       entiers naturels non nuls"""
    s = 0
    for k in range(n):
        x = 1/k
    s = s+x
    return s
```

Exercice 51 –

Question 1

Écrire une fonction `racine(n)` prenant en argument un entier naturel `n` et renvoyant sa racine carrée comme un entier si c'est un carré parfait, comme un flottant sinon.

Exercice 52 –

Question 1

Dans le jeu de la bataille navale, on représente chaque case par un couple d'entiers entre 0 et 9.

Un navire a ses extrémités sur les cases `a` et `b`. Un joueur tire sur la case `x`.

Écrire une fonction `touche(a,b,x)` qui renvoie un booléen indiquant si le navire est touché ou non.

Exercice 53 –

Question 1 *U*

n banquier vous propose un prêt de 400 000 euros sur 40 ans «à 3% par an» — ce qui, dans le langage commercial des banquiers, veut dire 0,25% par mois — avec des mensualités de 1431,93 euros. Autrement dit, vous contractez une dette de 400 000 euros. Chaque mois, cette dette augmente de 0,25% puis est diminuée du montant de votre mensualité. À la fin des 40×12 mensualités, il ne vous reste plus qu'à vous acquitter d'une toute petite dette, que vous rembourserez aussitôt.

a) Écrire une fonction `reste_a_payer(p, t, m, d)` renvoyant le montant de cette somme à rembourser immédiatement après le paiement de la dernière mensualité, où p est le montant total du prêt en euros (dans l'exemple, 400 000), t son taux mensuel (dans l'exemple, $0,25\times10^{-2}$), m le montant d'une mensualité en euros (dans l'exemple, 1431,93) et d la durée en années (dans l'exemple, 40).
Indice : dans le cas donné dans cet énoncé, vous devez trouver un montant restant d'un peu moins de 7,12 *euros.*

b) Écrire une fonction `somme_totale_payee(p, t, m, d)` renvoyant la somme totale (mensualités plus le dernier paiement) que vous aurez payé au banquier.

c) Écrire une fonction `cout_total(p, t, m, d)` renvoyant le coût total du crédit, c'est-à-dire le total de ce que vous avez payé moins le montant du prêt.

Exercice 54 –

Question 1 *U*

n banquier vous propose de vous prêter p euros, à un taux de $12t$% par an — ce qui, dans le langage commercial des banquiers, veut dire t% par mois — avec des mensualités de m euros. Autrement dit, vous contractez une dette de p euros. Chaque mois, cette dette augmente de t% puis est diminuée du montant de votre mensualité. Lorsque votre dette, augmentée du taux, est inférieure à la mensualité, il suffit de régler le solde en une fois.

Écrire une fonction `duree_mensualite(p,t,m)` renvoyant le nombre de mensualités nécessaires au remboursement total du prêt.

Attention : que se passe-t-il si la mensualité est trop petite ?

Indice : dans le cas où le prêt est $\mathtt{p}=4\times10^5$*, le taux est* $\mathtt{t}=0,25\times10^{-2}$ *et la mensualité est* $\mathtt{m}=1431,93$*, on trouvera une durée de remboursement de* 480 *mois.*

Exercice 55 –

Question 1

Écrire une fonction `comb(p,n)` renvoyant $\binom{n}{p}$ (nombre de combinaisons de p éléments parmi n). On pourra bien entendu introduire une fonction auxiliaire (c'est-à-dire, comme l'indique l'étymologie, une autre fonction dont le but sera de vous aider à répondre à la question).

Exercice 56 –

On appelle suite de Fibonacci la suite F définie par $F_0=0$, $F_1=1$ et pour tout $n\in\mathbb{N}$, $F_{n+2}=F_{n+1}+F_n$.

Question 1 *Écrire une fonction* `fib(n)` *calculant et renvoyant la valeur de* F_n.

Pensez à vérifier le résultat de votre fonction en 0, 1 et en 5 (vous calculerez à la main F_5 *avant* de le faire calculer par votre fonction).

Exercice 57 –

On pose $u_0=1$ et pour tout $n\in\mathbb{N}$,

$$u_{n+1}=\frac{1}{2}\left(u_n+\frac{n+1}{u_n}\right)$$

$$\text{et } v_n=\sum_{k=0}^{n}\frac{1}{u_k^5}$$

Question 1

Écrire une fonction `f(n)` renvoyant la valeur de v_n.

On peut montrer que $(v_n)_{n\in\mathbb{N}}$ converge.

Attention : on fera attention à ce que le calcul de `f` de demande pas trop de (re)calculs inutiles. Pour fixer les idées, vous pouvez considérer que `f(10**6)` doit être calculé en (largement) moins d'une minute.

Question 2

Vérifier que vous pouvez calculer v_n pour de grandes valeurs de n.

Exercice 58 –

Question 1

Écrire une fonction `somme1(n)` et une fonction `somme2(n)` prenant en argument un entier naturel `n` et renvoyant respectivement

$$\sum_{1\leq i,j\leq n} \frac{1}{i+j^2}, \tag{1}$$

$$\sum_{1\leq i<j\leq n} \frac{1}{i+j^2}. \tag{2}$$

Au besoin, on introduira des fonctions auxiliaires.

Exercice 59 –

On considère la suite u définie par $u_0 \in [-2;2]$ et $\forall n\in\mathbb{N}, u_{n+1}=\sqrt{2-u_n}$. On rappelle que u converge vers 1.

Question 1

Écrire une fonction `valeur_u(n,u0)` qui, à un entier naturel `n` et un flottant `u0`, renvoie u_n.

Question 2

Écrire une fonction `approche_u(eps,u0)` qui, à deux flottants `eps` et `u0`, renvoie le plus petit rang $n\in\mathbb{N}$ tel que $|u_n-1|\leq$ `eps`.

Exercice 60 –

Question 1

Écrire une fonction `comb(n,p)` calculant $\binom{n}{p}$ pour deux entiers naturels n,p vérifiant $0\leq p\leq n$, en utilisant la formule :

$$\binom{n}{p}=\frac{n}{p}\times\frac{n-1}{p-1}\times\cdots\times\frac{n-p+1}{1}.$$

On veillera à ce que le résultat donné par la fonction `comb` soit d'un type convenable.

Remarque : On ne demande pas de vérifier que les arguments de cette fonction vérifient les conditions imposées.

Question 2

Justifier que la fonction écrite donne bien le bon résultat, notamment à l'aide d'un invariant de boucle.

Exercice 61 –

Question 1 *Définir la fonction f qui à x associe*

$$\begin{cases} 2 & \text{si} \quad x\in[-4,-2] \\ -x & \text{si} \quad x\in[-2,0] \\ 0 & \text{si} \quad x\in[0,4] \end{cases}$$

Question 2 *Ecrire une fonction calculant le produit des entiers impairs de 1 à $2n+1$.*

a) Écrire une fonction `neg(b)` qui renvoie la négation du booléen `b` sans utiliser `not`.

b) Écrire une fonction `ou(a,b)` qui renvoie le ou logique des booléen `a` et `b` sans utiliser `not`, `or` ni `and`.

c) Écrire une fonction `et(a,b)` qui renvoie le et logique des booléen `a` et `b` sans utiliser `not`, `or` ni `and`.

Exercice 62 –

Écrire une fonction calculant le produit des entiers impairs de 1 à $2n+1$.

Exercice 63 –

Soit $(a,b,c)\in\mathbb{R}^2$, $a\neq 0$.

Écrire une fonction qui renvoie les solutions $ax^2+bx+c=0$ si celles-ci sont réelles, une phrase disant qu'il n'y a pas de solutions réelles sinon.

Modifier pour introduire le cas de la racine double.

Exercice 64 –

Écrire une fonction `somme_chiffres(n)` qui prend en argument un entier naturel `n` et qui renvoie la somme des chiffres de `n` (écrit en base dix).

On pourra utiliser toutes les fonctions de conversion présentes dans `Python`, mais la fonction rendue devra comporter au moins une boucle non triviale.

Exercice 65 –

Que renvoie la fonction suivante ? Le justifier, notamment à l'aide d'un invariant.

```
%[gobble=0,numbers=left]
def mystere(n,p):
    """Précondition : n entier positif, p entier
       """
    if p < 0 or p > n :
        return 0
    else :
        f = 1
        for i in range(p) :
            f = f * (n + 1 - p + i) // (i + 1)
        return f
```

Exercice 66 –

Pour tout $n\in\mathbb{N}^*$, on définit

$$H_n=\sum_{k=1}^{n}\frac{1}{k}.$$

Question 1

Écrire une fonction `H_depasse(M)` prenant en entrée un nombre `M` et renvoyant en sortie le plus petit entier naturel non nul n vérifiant $H_n\geq$ `M`.

Exercice 67 –

On considère la fonction suivante.

```
def mystere(L) :
    """Précondition : L est une liste de nombres
       """
    x,n,i = L[0],len(L),1
    while i<n and x > L[i] :
        L[i-1],L[i] = L[i],L[i-1]
        i = i+1
    return None
```

Question 1

Montrer que «`x` = `L[i-1]`» est un invariant de boucle pour la boucle `while` de la fonction `mystere`.

Question 2

Donner un variant de boucle pour la boucle `while` de la fonction `mystere`. Que peut-on en déduire ?

Question 3

Si `L` est une liste de nombres, que fait `mystere(L)` ? Le justifier, notamment à l'aide des questions précédentes (vous pourrez cependant écrire un ou plusieurs autres invariants, au besoin).

Modéliser

Remarque : vous avez tout intérêt à utiliser cette fonction et à observer son fonctionnement *avant* de répondre à ces questions.

Exercice 68 –

On considère la fonction suivante.

```
def mystere(a,b) :
    """Précondition : a,b sont des entiers, a>0,
        b>1"""
    k,p = 0,1
    while a % p == 0 :
        k = k+1
        p = p*b
    return k-1
```

Question 1
Dresser un tableau de valeurs décrivant les valeurs des variables `k` et `p` en entrée des trois premiers tours de la boucle `while` de la fonction `mystere(a,b)`.

On pourra au besoin faire intervenir les variables `a` et `b`.

Question 2
En s'aidant de la question précédente, écrire un invariant de boucle pour la boucle `while` de la fonction `mystere(a,b)`. On justifiera la réponse.

Question 3
Donner un variant de boucle pour la boucle `while` de la fonction `mystere(a,b)`. On justifiera la réponse.

Question 4
Déduire des questions précédentes qu'un appel de la fonction `mystere(a,b)` renvoie un résultat et déterminer le résultat alors renvoyé. On justifiera la réponse.

Exercice 69 –

Question 1 *Calculer* 2^9 *à l'aide d'une boucle itérative.*

Question 2 *Écrire un algorithme affichant la table de multiplication de 9.*

Question 3 *Calculer* $16!$ *à l'aide d'une boucle itérative.*

Question 4 *Calculer*

$$\sum_{k=0}^{15} \frac{1}{k!}$$

Question 5 *Écrire une fonction calculant le nombre de chiffres d'un entier écrit en base 10.*

Question 6 *On considère la suite* u *définie par* $\forall n \in \mathbb{N}^* \quad u_n = \sum_{k=1}^{n} \frac{1}{\sqrt{k}}$ *Quel est la plus petite valeur* n *pour laquelle* $u_n \geq 1000$ *?h*

Question 7 *Écrire une fonction trouvant le plus petit nombre premier supérieur ou égal à un entier donné.*

Question 8 *Écrire une fonction calculant le nombre de diviseurs d'un entier* n *donné.*

Question 9 *Calculer* p_5/q_5 *où* p *et* q *sont définies par :*

$$\begin{aligned} p_0 &= 1 \\ q_0 &= 1 \\ \forall n \in \mathbb{N} \quad p_{n+1} &= p_n^2 + 2q_n^2 \\ \forall n \in \mathbb{N} \quad q_{n+1} &= 2p_n q_n \end{aligned}$$

Exercice 70 –

Un banquier vous propose un prêt de 400 000 euros sur 40 ans «à 3% par an» — ce qui, dans le langage commercial des banquiers, veut dire 0,25% par mois — avec des mensualités de 1431,93 euros. Autrement dit, vous contractez une dette de 400 000 euros. Chaque mois, cette dette augmente de 0,25% puis est diminuée du montant de votre mensualité. À la fin des 40×12 mensualités, il ne vous reste plus qu'à vous acquitter d'une toute petite dette, que vous rembourserez aussitôt.

Question 1
Écrire une fonction `reste_a_payer(p, t, m, d)` renvoyant le montant de cette somme à rembourser immédiatement après le paiement de la dernière mensualité, où p est le montant total du prêt en euros (dans l'exemple, 400 000), t son taux mensuel (dans l'exemple, $0,25 \times 10^{-2}$), m le montant d'une mensualité en euros (dans l'exemple, 1431,93) et d la durée en années (dans l'exemple, 40).

Indice : dans le cas donné dans cet énoncé, vous devez trouver un montant restant d'un peu moins de 7,12 *euros.*

Question 2
Écrire une fonction `somme_totale_payee(p, t, m, d)` renvoyant la somme totale (mensualités plus le dernier paiement) que vous aurez payé au banquier.

Question 3
Écrire une fonction `cout_total(p, t, m, d)` renvoyant le coût total du crédit, c'est-à-dire le total de ce que vous avez payé moins le montant du prêt.

Un banquier vous propose de vous prêter p euros, à un taux de $12t$% par an — ce qui, dans le langage commercial des banquiers, veut dire t% par mois — avec des mensualités de m euros. Autrement dit, vous contractez une dette de p euros. Chaque mois, cette dette augmente de t% puis est diminuée du montant de votre mensualité. Lorsque votre dette, augmentée du taux, est inférieure à la mensualité, il suffit de régler le solde en une fois.

Question 4
Écrire une fonction `duree_mensualite(p,t,m)` renvoyant le nombre de mensualités nécessaires au remboursement total du prêt.

Question 5
Attention : que se passe-t-il si la mensualité est trop petite?

Indice : dans le cas où le prêt est $\mathtt{p} = 4 \times 10^5$*, le taux est* $\mathtt{t} = 0,25 \times 10^{-2}$ *et la mensualité est* $\mathtt{m} = 1431,93$*, on trouvera une durée de remboursement de* 480 *mois.*

Question 6
Écrire une fonction `tracer_mensualite(p,t,m)` permettant de tracer en fonction du numéro de la mensualité la dette restante (ou le capital restant dû) jusqu'à ce que le prêt soit remboursé. Cette fonction permettra également de tracer en fonction du numéro de la mensualité le montant de l'intérêt versé à la banque.

2 Algorithmique

Exercice 71 –

Votre robot dispose de nombreux récepteurs et enregistre tous les signaux qui l'entourent. Cependant vous avez remarqué que certains de ces signaux sont très bruités. Vous décidez donc d'écrire un programme qui atténue le bruit de ces signaux, en effectuant ce que l'on appelle un lissage.

Une opération de lissage d'une séquence de mesures (des nombres décimaux) consiste à remplacer chaque mesure sauf la première et la dernière, par la moyenne des deux valeurs qui l'entourent.

Par exemple, si l'on part de la séquence de mesures suivantes : 1 3 4 5

On obtient après un lissage : 1 2.5 4 5

Le premier et dernier nombre sont inchangés. Le deuxième nombre est remplacé par la moyenne du 1er et du 3e, soit $(1+4)/2 = 2.5$, et le troisième est remplacé par $(3+5)/2 = 4$.

On peut ensuite repartir de cette nouvelle séquence, et refaire un nouveau lissage, puis un autre sur le résultat, etc. Votre programme doit calculer le nombre minimum de lissages successifs nécessaires pour s'assurer que la valeur absolue de la différence entre deux valeurs successives de la séquence finale obtenue ne dépasse jamais une valeur donnée, `diffMax`.

On vous garantit qu'il est toujours possible d'obtenir la propriété voulue en moins de 5000 lissages successifs.

On cherche à définir la fonction suivante `def lissage(t:list[float], diffmax:float) -> int :` où

- l'entrée est une liste `t` contenant les mesures, qui sont des flottants, et un flottant `diffmax` ;
- la sortie est un entier qui correspond au nombre minimal de lissages nécessaire.

▪ **Exemple** `lissage([1.292, 1.343, 3.322, 4.789, -0.782, 7.313, 4.212], 1.120) -> 13.` ▪

Exercice 72 –

Il existe de nombreuses traditions étranges et amusantes sur Algoréa, la grande course de grenouilles annuelle en fait partie. Il faut savoir que les grenouilles algoréennes sont beaucoup plus intelligentes que les grenouilles terrestres et peuvent très bien être dressées pour participer à des courses. Chaque candidat a ainsi entraîné sa grenouille durement toute l'année pour ce grand événement.

La course se déroule en tours et, à chaque tour, une question est posée aux dresseurs. Le premier qui trouve la réponse gagne le droit d'ordonner à sa grenouille de faire un bond. Dans les règles de la course de grenouilles algoréennes, il est stipulé que c'est la grenouille qui restera le plus longtemps en tête qui remportera la victoire. Comme cette propriété est un peu difficile à vérifier, le jury demande votre aide.

Ce que doit faire votre programme :

`nbg` numérotées de 1 à `nbg` sont placées sur une ligne de départ. À chaque tour, on vous indique le numéro de la seule grenouille qui va sauter lors de ce tour, et la distance qu'elle va parcourir en direction de la ligne d'arrivée.

Écrivez un programme qui détermine laquelle des grenouilles a été strictement en tête de la course à la fin du plus grand nombre de tours.

ENTRÉE : deux entiers `nbg` et `nbt` et un tableau `t`.
`nbg` est le nombre de grenouilles participantes.
`nbt` est le nombre de tours de la course.
`t` est un tableau ayant `nbt` éléments, et tel que chaque élément est un couple : (numéro de la grenouille qui saute lors de ce tour, longueur de son saut).
SORTIE : vous devez renvoyer un entier : le numéro de la grenouille qui a été strictement en tête à la fin du plus grand nombre de tours. En cas d'égalité entre plusieurs grenouilles, choisissez celle dont le numéro est le plus petit.

EXEMPLE :
entrée : (4, 6, [[2,2], [1,2], [3,3], [4,1], [2,2], [3,1]])
sortie : 2.

Exercice 73 –

Un marchand de légumes très maniaque souhaite ranger ses petits pois en les regroupant en boîtes de telle sorte que chaque boîte contienne un nombre factoriel de petits pois. On rappelle qu'un nombre est factoriel s'il est de la forme 1, 1×2, $1\times 2\times 3$, $1\times 2\times 3\times 4\cdots$ et qu'on les note sous la forme suivante :

$$n! = 1\times 2\times 3\times \cdots \times (n-1)\times n$$

Il souhaite également utiliser le plus petit nombre de boîtes possible.
Ainsi, s'il a 17 petits pois, il utilisera :

2 boîtes de $3! = 6$ petits pois
2 boîtes de $2! = 2$ petits pois
1 boîte de $1! = 1$ petits pois.

ce qui donne bien $2\times 3! + 2\times 2! + 1\times 1! = 12+4+1 = 17$.

D'une manière générale, s'il a `nbp` petits pois, il doit trouver une suite `a_1,a_2⋯,a_p` d'entiers positifs ou nuls avec `a_p`> 0 et telle que : `nbp=a_1 x 1!+a_2 x 2!+⋯+a_p x p!` avec `a_1+⋯+a_p` minimal.
Remarque mathématique : si à chaque étape on cherche le plus grand entier `k` possible tel que `k!` soit inférieur au nombre de petits pois restant, on est sûrs d'obtenir la décomposition optimale : en termes informatiques, on dit que l'algorithme *glouton* est optimal.

ENTRÉE : un entier, `nbp`, le nombre total de petits poids.
SORTIE : un couple constitué de l'entier `p` et du tableau `[ a_1, a_2, ⋯, a_p ]`.

EXEMPLE :
entrée : 17
sortie : `(3, [ 1, 2, 2 ] )`.

Exercice 74 –

Rien de tel que de faire du camping pour profiter de la nature. Cependant sur Algoréa, les moustiques sont particulièrement pénibles et il faut faire attention à l'endroit

Modéliser

où l'on s'installe, si l'on ne veut pas être sans cesse piqué.

Vous disposez d'une carte sur laquelle est indiquée, pour chaque parcelle de terrain, si le nombre de moustiques est supportable ou non. Votre objectif est de trouver le plus grand camping carré évitant les zones à moustiques qu'il est possible de construire.

ENTRÉE : un tableau `t` de n éléments correspondant à des lignes, chacune de ces lignes contenant p élements, qui ne sont que des 0 et des 1. 0 signifie qu'il n'y a pas de moustiques et 1 qu'il y a des moustiques.
SORTIE : un entier : la taille maximale du côté d'un carré ne comportant que des 0 et dont les bords sont parallèles aux axes.

EXEMPLE 1 :

entrée : `t`=
```
[ [1, 0, 0, 1, 0, 0, 1],
  [0, 0, 0, 0, 0, 0, 0],
  [1, 0, 0, 0, 0, 0, 0],
  [0, 0, 0, 0, 0, 0, 0],
  [0, 1, 0, 0, 0, 0, 1],
  [1, 0, 0, 0, 1, 0, 1] ]
```
sortie : 4.

EXEMPLE 2 :

entrée : `t`=
```
[ [0, 0, 0, 1, 1, 1, 1],
  [0, 0, 0, 1, 1, 1, 1],
  [0, 0, 0, 1, 1, 1, 1],
  [1, 1, 1, 1, 1, 0, 0],
  [1, 1, 1, 1, 1, 0, 0] ]
```
sortie : 3.

Exercice 75 –

On ne s'intéresse pas ici à la validité d'un nombre écrit en chiffre romains, mais à sa valeur. On rappelle quelques principes de base. Les sept caractères de la numération romaine sont :

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Certaines lettres sont dites d'*unité*. Ainsi on dit que I est une unité pour V et X, X est une unité pour L et C, C est une unité pour D et M.

Pour trouver la valeur d'un nombre écrit en chiffres romains, on s'appuie sur les règles suivantes :

- toute lettre placée à la droite d'une lettre dont valeur est supérieure ou égale à la sienne s'ajoute à celle-ci ;
- toute lettre d'unité placée immédiatement à la gauche d'une lettre plus forte qu'elle indique que le nombre qui lui correspond doit être retranché au nombre qui suit ;
- les valeurs sont groupées en ordre décroissant, sauf pour les valeurs à retrancher selon la règle précédente ; 1
- la même lettre ne peut pas être employée quatre fois consécutivement sauf M.

Par exemple, DXXXVI = 536, CIX = 109 et MCMXL = 1940.

1. Écrire une fonction `valeur (caractere)` qui retourne la valeur décimale d'un caractère romain. Cette fonction doit renvoyer 0 si le caractère n'est pas l'un des 7 chiffres romains.
2. Écrire la fonction principale `conversion (romain)` qui permet de convertir un nombre romain en nombre décimal. Cette fonction doit prendre en argument une chaîne de caractères `romain`. Si cette chaîne est écrite en majuscule et correspond à un nombre romain correctement écrit, la fonction doit renvoyer le nombre décimal égal au nombre romain passé en argument. Sinon, la fonction doit renvoyer -1.

Exercice 76 –

Pour tout $n \in \mathbb{N}\setminus\{0,1\}$, on pose $S_n = \sum_{k=2}^{n} \frac{(-1)^k}{k \ln k}$. On admet que la suite $(S_n)_{n\in\mathbb{N}\setminus\{0,1\}}$ a une limite finie ℓ en $+\infty$, et que pour tout $n \in \mathbb{N}\setminus\{0,1\}$, $u_{2n+1} \leq \ell \leq u_{2n}$.

1. Écrire un script Python donnant une valeur approchée de ℓ à 10^{-8} près.
2. Démontrer que le résultat donné par cet script est correct. On donnera clairement les éventuels invariants et variants des boucles intervenant dans le programme.

Exercice 77 –

On appelle *nombre parfait* tout entier naturel non nul qui est égal à la somme de ses diviseurs stricts, c'est-à-dire de ses diviseurs autres que lui-même.

Par exemple, 26 n'est pas parfait, car ses diviseurs stricts sont 1, 2 et 13, et $1+2+13 = 16 \neq 28$. Mais 28 est parfait, car ses diviseurs stricts sont 1, 2, 4, 7 et 14, et $1+2+4+7+14 = 28$.

Question 1

Écrire une fonction Python `parfait(n)` prenant en entrée un entier naturel non nul `n`, et renvoyant un booléen donnant la valeur de vérité de l'assertion « `n` est parfait ».

Question 2

Écrire les éventuels variants et invariants permettant de montrer que cette fonction renvoie le bon résultat.

Exercice 78 –

Question 1

Écrire un script `Python` permettant de calculer le plus petit entier naturel n tel que $n! > 123456789$.

Question 2

Écrire les éventuels variants et invariants de boucle permettant de montrer que le code `Python` écrit précédemment donne le bon résultat.

Question 3

Montrer que les variants et/ou invariants donnés à la question précédente sont bien des variants/invariants de boucle et justifier que le script écrit termine et donne le bon résultat.

Exercice 79 –

On s'intéresse au problème du codage d'une suite de bits (représentée sous la forme d'un tableau de 0 ou 1), de manière à pouvoir réparer une erreur de transmission. On fixe un entier naturel non nul `k`. Un tableau de bits `b` sera codé avec un niveau de redondance `k` en répétant chaque bit $2k+1$ fois. Pour décoder un tableau avec un niveau de redondance `k`, on le découpe en blocs de $2k+1$ bits. Dans chaque bloc, on effectue un « vote » et l'on considère la valeur majoritaire.

Exemple : Avec $k=2$ (et donc un niveau de redondance de 2), le tableau

$$b=[0,1,0]$$

sera codé en

$$c=[\underbrace{0,0,0,0,0}_{\text{5 bits}},\underbrace{1,1,1,1,1}_{\text{5 bits}},\underbrace{0,0,0,0,0}_{\text{5 bits}}].$$

Imaginons qu'après transmission, le tableau reçu soit

$$c'=[\underbrace{0,0,0,1,0}_{\text{1 erreur}},\underbrace{0,1,1,0,1}_{\text{2 erreurs}},\underbrace{0,1,0,1,1}_{\text{3 erreurs}}].$$

On le décode alors en

$$b'=[0,1,1].$$

Question 1
Écrire une fonction `code(b,k)` renvoyant le tableau codant un tableau `b`, avec un niveau de redondance `k`.

Question 2
Écrire une fonction `decode(c,k)` renvoyant le tableau décodant un tableau `c`, avec un niveau de redondance `k`.

Exercice 80 –

On considère un fichier `points.csv` contenant n lignes, chaque ligne contenant n entiers parmi 0 ou 1, séparés par des virgules. Ce fichier représente donc un tableau de nombres. Par exemple, le fichier

```
1,1,0,1
1,0,1,0
0,0,0,0
```

sera représenté (en Python) par le tableau bidimensionnel `t` (construit comme un tableau de tableaux) :

```
[1,1,0,1],
[1,0,1,0],
[0,0,0,0] ]
```

Question 1
Écrire une fonction `lit_fichier(nom_de_fichier)` qui, à un tel fichier `nom_de_fichier`, renvoie le tableau associé.

On voit ce tableau comme décrivant des points dans le plan. Étant donné un tel tableau `t`, on considère que l'on a un point aux coordonnées `(i,j)` si et seulement si `t[i][j]` vaut 1. Par exemple, avec le tableau précédents, la liste `L` des points décrits est

```
L = [ (0,1) , (1,0) , (1,1) , (1,3) , (2,0) ,
    (2,2) ]
```

Question 2
Écrire une fonction `lit_tableau(t)` qui, à un tel tableau bidimensionnel `t`, renvoie la liste des points décrits.

Question 3
Complexité ou invariants ?

Question 4
Écrire une fonction `d(a,b)` qui, pour deux couples d'entiers `a`, `b`, dont nous noterons les coordonnées repectivement (x_a,y_a) et (x_b,y_b), renvoie la valeur $(x_a-x_b)^2+(y_a-y_b)^2$.

On veut maintenant, étant donné un entier naturel k non nul et un couple d'entiers `c`, trouver les k couples de la liste de points les plus proches de `c`. S'il y a égalité entre plusieurs points, on n'en garde que k.

Par exemple [...]

On considère le code suivant.

```
def kNN(L,c,k,d):
    """k plus proches voisins du point c dans L
       d : fonction de distance"""
    v = []
    for j in range(L) :
        a = L[j]
        if len(v) < k :
            v.append(a)
        if d(a,c) < d(v[-1],c) :
            v[-1] = a
        i = len(v) - 1
        while i >= 1 and v[i] < v[i-1] :
            v[i-1],v[i] = v[i],v[i-1]
            i = i-1
    return v
```

Question 5
Donner l'invariant pour la boucle while et faire montrer que c'en est un.

Question 6
Donner l'invariant pour la boucle for.

Question 7
Complexité.

Exercice 81 –

On s'intéresse au problème d'insertion d'un nombre dans un tableau de nombres trié par ordre croissant.

Soit `t` $=[t_0,\dots,t_{n-1}]$ un tableau de nombres trié par ordre croissant, c'est-à-dire que

$$t_0\le t_1\le\cdots\le t_{n-1}.$$

On dit qu'un nombre x s'insère en position $i\in[0,n-1]$ dans le tableau `t` si $t_i\le x\le t_{i+1}$. Si $x<t_0$, alors x s'insère en position -1 dans `t` et, si $x>t_{n-1}$, alors x s'insère en position $n-1$ dans `t`.

Question 1
Écrire une fonction `position_insertion(t,x)` prenant en argument un tableau de nombres `t` trié par ordre croissant et un nombre `x` et renvoyant une position où `x` s'insère dans `t`.

Exercice 82 –

Soit $n\ge 2$ un entier. Un diviseur strict de n est un entier $1\le d\le n-1$ qui divise n (c'est-à-dire que le reste de la division euclidienne de n par d est nul).

Deux entiers n_1 et n_2 sont dits *amicaux* si la somme des diviseurs stricts de n_1 vaut n_2 et si la somme des diviseurs stricts de n_2 vaut n_1.

Question 1
Écrire une fonction `amicaux(n,m)` qui prend en argument deux entiers naturels `n` et `m` et renvoie la valeur de vérité de « `n` et `m` sont amicaux ».

On pourra écrire une fonction auxiliaire, au besoin.

Exercice 83 –

La société Sharp commercialise des caisses automatiques utilisées par exemple dans des boulangeries. Le client glisse directement les billets ou les pièces dans la machine qui se charge de rendre automatiquement la monnaie.

Objectif Afin de satisfaire les clients, on cherche à déterminer un algorithme qui va permettre de rendre le moins de monnaie possible.

La machine dispose de billets de 20€, 10€ et 5€ ainsi que des pièces de 2€, 1€, 50, 20, 10, 5, 2 et 1 centimes.

On se propose donc de concevoir un algorithme qui demande à l'utilisateur du programme la somme totale à payer ainsi que le montant donné par l'acheteur. L'algorithme doit alors afficher quels sont les billets et les pièces à rendre par le vendeur.

Question 1
Mettre en place une structure de liste pour gérer les valeurs des billets ou des pièces et la nommer `valeurs`.

Question 2
Écrire une fonction `rendre_monnaie(cout,somme_client,valeurs)` prenant en arguments deux flottants `cout` et `somme_client` représentant le coût d'un produit et la somme donnée par le client en € ainsi que `valeurs`. Cette fonction renverra une liste `nombre_billets` qui donnera pour chaque terme de `valeurs` le nombre de pièces et/ou billets à rendre.

Question 3
Créer une fonction `afficher_rendu_monnaie(cout,somme_client,valeurs)` permettant d'afficher le nombre de pièces et billets à rendre comme l'exemple ci-dessous.

```
afficher_rendu_monnaie(15.99,17.5,valeurs)

0 : billet de 20 euros
0 : billet de 10 euros
0 : billet de 5 euros
0 : pièce de 2 euros
1 : pièce de 1 euros
1 : pièce de 50 centimes
0 : pièce de 20 centimes
0 : pièce de 10 centimes
0 : pièce de 5 centimes
0 : pièce de 2 centimes
1 : pièce de 1 centimes
```

Exercice 84 –

On donne la `fonctionMystere(n)` définie comme suit.

```
def fonctionMystere(n) :
    if n==0 or n==1:
        return 1
    else :
        res = 1
    for i in range (2,n+1) :
        res = res * i
    return res
```

Question 1 *Si $n = 5$ quelles sont les valeurs que va prendre la variable* `i` *?*

Question 2 *Si $n = 4$ donner les valeurs successives que vont prendre les variables* `i` *et* `res` *lorsqu'on exécute l'algorithme.*

Question 3 *Quel est le nom mathématique usuel donné à la fonction* `fonctionMystere` *?*

Exercice 85 –

```
def fonction(x) :
    y=1.1
    i=0
    while x!=y and i <10:
        x=x+0.1
        i=i+1
    return i
```

Question 1 *On exécute l'instruction* `fonction(0.1)`. *À chaque itération, donner la valeur de i et de x.*

Question 2 *On exécute l'instruction* `fonction(0.3)`. *À chaque itération, donner la valeur de i et de x.*

Exercice 86 –

Question 1 *Écrire une fonction* `ajouteUnFor(L)` *qui prend comme argument une liste* `L` *de flottants et qui ajoute 1 à chaque élément de la liste. On utilisera une boucle* ***for****.*

Question 2 *Écrire une fonction* `ajouteUnWhile(L)` *qui prend comme argument une liste* `L` *de flottants et qui ajoute 1 à chaque élément de la liste. On utilisera une boucle* ***while****.*

Question 3 *Expliquer pourquoi il n'est pas indispensable que la fonction renvoie la liste modifiée.*

Exercice 87 –

Pour les deux questions suivantes, les fonctions `max` et `min` ne sont pas autorisées.

Question 1 *Écrire une fonction* `chercheMax(L)` *qui prend comme argument une liste* `L` *d'entiers* `int` *et qui renvoie le plus grand élément de la liste.*

Question 2 *Écrire une fonction* `chercheMaxIndice(L)` *qui prend comme argument une liste* `L` *d'entiers* `int` *et qui renvoie l'indice du plus grand élément de la liste.*

Exercice 88 –

On veut tester si un entier `n` est premier on donne l'algorithme suivant :

```
def est_premier(n):
    """ Renvoie True si n est premier, False
        sinon
        Préconditon : n est un entier."""
    for d in range(2,n):
        if n % d == 0:
            return False
    return True
```

Question 1 *Proposer un invariant de boucle pour démontrer cet algorithme.*

Exercice 89 –

Question 1

Donnons ces invariant et variant pour l'algorithme de vérification de la conjecture de syracuse (nous ne pouvons malheureusement pas le faire pour l'exemple **??**, puisque comme son nom l'indique, la conjecture de Syracuse n'a jamais été demontrée).

Conjecture de Syracuse : on note $f : \mathbb{N}^* \to \mathbb{N}^*$ l'application vérifiant, pour tout n pair $f(n) = n/2$ et tout n impair et $f(n) = 3n+1$.

On conjecture que pour tout entier n, il existe k tel que $f^k(n) = 1$.

Voici un algorithme calculant, pour tout n donné, le plus petit entier k vérifiant $f^k(n) = 1$:

```
def f(n):
    """Fonction de Syracuse.
    Précondition : n est un entier strictement
        positif"""
    if n % 2 == 0:
        return n // 2
    else:
        return 3 * n + 1

def syracuse(n):
    """Renvoie le premier entier k tel que f\^k(
        n) = 0.
    Précondition : n est un entier strictement
        positif"""
    x = n
    k = 0
    while x != 1:
        x = f(x)
        k = k + 1
    return k
```

Exercice 90 –

Commencez par recopier le code suivant dans votre script.

```
def binaire(k,n):
    """Renvoie le tableau de n bits écrivant k
        en binaire
    Précondition : 0 <= k <= 2**n -1 """
    L = [0]*n
    p = k
    for i in range(n):
        L[n-1-i] = p % 2
        p = p // 2
    return L
```

Soit `k` un entier écrit en binaire avec `n` chiffres : $\mathtt{k} = \underline{a_{n-1}\dots a_1 a_0}_2$, c'est-à-dire que

$$\mathtt{k} = \sum_{j=0}^{n-1} a_j 2^j.$$

Question 1

Écrire un invariant portant sur `L` et `p` dans la boucle `for` de la fonction `binaire(k,n)` et justifier que cette fonction renvoie bien le résultat demandé.

Exercice 91 –

Question 1 *Soit les algorithmes de calculs de moyenne ci-dessous, proposez des invariants de boucles.*

```
def moyenne(t):
    """Calcule la moyenne de t
       Précondition : t est un tableau de
       nombres non vide"""
    s = 0
    for x in t:
        s = s + x
    return s/len(t)

def moyenne(t):
    """Calcule la moyenne de t
       Précondition : t est un tableau de
       nombres non vide"""
    n = len(t) # Longueur de t
    s = 0
    for i in range(n):
        s = s + t[i]
    return s/n
```

Question 2 *Soit l'algorithme de calculs de variance ci-dessous, proposez un invariants de boucles.*

```
def variance(t):
    """Renvoie la variance de t
       Précondition : t est un tableau de
       nombres non vide"""
    sc = 0
    for x in t:
        sc = sc + x**2
    return sc/len(t) - moyenne(t)**2
```

Exercice 92 –

Question 1 *Soit les algorithmes de recherche de maximum d'un tableau ci-dessous, proposez des invariants de boucles.*

```
def maxi(t):
    """Renvoie le plus grand élément de t.
       Précondition : t est un tableau
       non vide"""
    m = t[0]
    for x in t:
        if x > m:
            m = x
    return m

def maxi(t):
    """Renvoie le plus grand élément de t.
       Précondition : t est un tableau
       non vide"""
    m = t[0]
    for i in range(1, len(t)):
        if t[i] > m:
            m = t[i]
    return m
```

Question 2 *Soit les algorithmes de recherche d'indice de maximum d'un tableau ci-dessous, proposez des invariants de boucles.*

```
def indicemaxi(t):
```

```
    """Renvoie l'indice du plus grand
        élément de t.
       Précondition : t est un tableau
       non vide"""
    im = 0
    for i in range(1, len(t)):
        if t[i] > t[im]:
            im = i
    return im

def indicemaxi(t):
    """Renvoie l'indice du plus grand élément
       de t.
       Précondition : t est un tableau
       non vide"""
    im = 0
    for i, x in enumerate(t):
        if x > t[im]:
            im = i
    return im
```

Exercice 93 –

Question 1 *Soit les algorithmes de test d'appartenance d'un élément dans un tableau. Proposer un invariant de boucle.*

```
def appartient(e, t):
    """Renvoie un booléen disant si e
    appartient àt
       Précondition : t est un tableau"""
    for x in t:
        if e == x:
            return True
    return False
```

Question 2 *Soit les algorithmes de recherche d'indice de première occurrence d'un élément dans un tableau. Proposer un invariant de boucle.*

```
def ind_appartient(e,t):
    """Renvoie l'indice de la première
       occurrence de e dans t,
       None si e n'est pas dans t
       Précondition : t est un tableau"""
    for i in len(t):
        if t[i] == e:
            return i
    return None
```

Introduction

L'analyse harmonique (fréquentielle) des systèmes permet de mettre en évidence de nombreuses caractéristiques telles que la bande passante, la fréquence de coupure, la résonance, etc. Elle sera aussi très utile pour étudier la stabilité d'un système en 2nde année.

Un système modélisé linéaire peut se caractériser dans le domaine symbolique de Laplace par sa fonction de transfert $H(p)$. Sa forme générale est :

$H(p) = \frac{a_m p^m + \ldots + a_1 p + a_0}{b_n p^n + \ldots + b_1 p + b_0}$ où les coefficients $a_i, i \in [0;m]$ et $b_k, k \in [0;n]$ sont réels.

La variable p est un nombre complexe. Pour l'analyse fréquentielle, on pose $p = j\omega$ ($p \in \mathbb{C}$). C'est le cas particulier de la transformée de Fourier. On parle alors aussi de transmittance $H(j\omega)$.

Une représentation classique de cette fonction, est le diagramme de Bode (figure **??**). Il fait apparaître deux quantités : le gain (dB) et la phase (°).

> *L'objectif des questions qui suivent est d'extraire certaines propriétés d'un système linéaire à partir de sa fonction de transfert.*

Diagramme de Bode

Le tracé de la figure **??** a été obtenu à l'aide du script donné en annexe 1.

Question 3

En analysant le script donné en annexe 1, donner les fonctions de transfert dont le diagramme de Bode est représenté sur la figure **??** ?

Question 4

A une étape de l'algorithme proposé,

1. combien de données contiennent les listes `w`, `gain` et `phase` ?
2. Donner le nombre de bits nécessaire au codage des flottant en double précision.
3. En admettant que chacune de ces données est de type `float` codé en double précision, quelle quantité de mémoire est nécessaire pour le stockage de ces trois listes ?

Propriétés caractéristiques du système

Asymptote infinie de la courbe en gain

Trois listes de même dimension `w`, `gain` et `phase` contiennent les données permettant le tracé. On souhaite déterminer l'asymptote lorsque $\omega \to \infty$ de la courbe de gain de la figure 1.

Question 5

Proposer une instruction permettant de donner la valeur de l'asymptote lorsque $\omega \to \infty$ de la courbe de gain en dB/decade (décibels par décade).

Résonance

Les courbes en **gain** de la figure **??** présentent **un maximum** appelé « pic de résonance ». En effet, lorsque le gain est positif, c'est qu'il y a amplification du signal d'entrée.

Dans le contexte de la figure **??** (au maximum un seul pic de réconance), on donne ci-dessous la fonction `picResonance(w,gain,phase)` qui retourne pour une courbe un triplet de valeurs respectives la pulsation de résonance *wr*, le gain maximal *gr* et la phase correspondante *pr* : `(wr,gr,pr)`.

```
def picResonance(w,gain,phase):
    n=len(w)
    gr=gain[0]
    i=1
    while i<n and gr<=gain[i]:
        gr=gain[i]
        i+=1
    if i==1:
        return ()
```

```
    else:
        return (w[i-1],gr,phase[i-1])
```

Question 6

Pour la fonction proposée (`picResonance(w,gain,phase)`), proposer un invariant de boucle. Dans le cas où ce pic n'existerait pas, que renvoie la fonction? Proposer un variant de boucle et démontrer que l'algorithme renvoie bien un résultat.

Dans certains cas, le système peut présenter plusieurs pics de résonance (figure **??**).

Question 7

Dans le cas où le diagramme en gain serait multi-pics, écrire la fonction `picsResonance(w,gain,phase)` retournant une liste `L` de triplets de valeurs respectives une pulsation de résonance `wr`, le gain `gr` et la phase correspondante `pr` : `(wr,gr,pr)`. La liste `L` peut présenter l'allure suivante : `[(wr1,gr1,pr1),(wr2,gr2,pr2),...,(wrk,grk,prk)]`. Dans le cas où il n'y aurait aucun pics, la fonction retourne une liste vide.

Bande passante

Pour un filtre passe-bas, la bande passante peut être définie comme la plage de pulsations $\omega \in]0;\omega_c]$ pour lesquelles le gain est supérieur ou égal à $G_{\max} - 3\text{dB}$. On peut ainsi préciser la pulsation de coupure ω_c.

Dans tout ce qui suit, on se place dans le cas d'un filtre passe-bas passif ($G_{\text{dB}}(0) = 0$) non résonant. La fonction G_{dB} est monotone décroissante et l'équation $G_{\text{dB}}(\omega) + 3 = 0$ admet toujours une unique solution. On est par exemple dans le cas de la figure 3. Dans la bande passante, l'atténuation du signal d'entrée ne dépasse pas alors environ 30 %.

Question 8

Écrire la fonction `pulsationCoupure(w,gain)` retournant la valeur de la pulsation de coupure `wc` en utilisant une méthode par balayage de la liste `gain`. On s'assurera de la terminaison de l'algorithme. Si la pulsation de coupure n'existe pas, on retourne -1.

La méthode de dichotomie pour résoudre une équation est basée sur le théorème des valeurs intermédiaires. Soit $f : [a,b] \to \mathbb{R}$ une fonction continue, alors f prend toutes les valeurs intermédiaires entre $f(a)$ et $f(b)$. En particulier, si f est telle que $f(a) \times f(b) < 0$, alors il existe $\alpha \in]a,b[$ tel que $f(\alpha) = 0$.

Question 9

Proposer une fonction `pulsationCoupure(w,gain)` retournant la valeur de la pulsation de coupure `wc` en utilisant une méthode de dichotomie sur la liste `gain`. Si la pulsation de coupure n'existe pas, on retourne -1.

Stockage des données dans un fichier texte

On souhaite stocker le contenu des listes `w`, `gain` et `phase` dans un fichier texte `"bode.txt"`.

L'annexe 2 donne quelques fonctions python. Par exemple, pour ouvrir un fichier en écriture, on peut écrire `f = open("bode.txt","w")`. La variable `f` est alors un objet de type fichier.

La structure attendu dans le fichier texte `"bode.txt"` est la suivante (cas de la figure 2) :

```
pulsation;gain;phase
0.1;0.290546581842;-0.306830090606
0.12;0.421009962397;-0.373102215459
0.14;0.577344844946;-0.442256557908
...
```

R La première ligne permet de préciser le type des données du fichier. La suite comporte autant de lignes que de données. Les séparateurs sont des points virgules ";".

Question 10

1. Quelle sera la taille approximative du fichier texte `"bode.txt"` sachant que le nombre de données est identique à celui de la question 2 et que l'on suppose les caractères sont codés en ASCII (1 caractère sur un octet)?
2. Quel serait le gain de taille en % si on se limitait à 3 chiffres significatifs pour chacune des valeurs numériques?

Question 11

Écrire un script utilisant les listes `w`, `gain` et `phase`, et permettant de créer le fichier "bode.txt". A la fin de l'écriture, on assurera sa fermeture « propre ».

Question 12

Proposer une fonction `retourneListes(nomFichier)` prenant en argument une chaîne de caractère `nomFichier` et retournant les listes `w` (pulsations) et `gain` (gains). A la fin de la lecture, on assurera la fermeture « propre » du fichier.

– Fin de sujet –

Annexe 1 : Diagrammes de Bode

```
import numpy as np
from scipy import signal
from matplotlib import pyplot as plt

# Fonction de transfert sous la forme num(p)/
    den(p)
# Coefficients au numerateur
# du plus grand ordre au plus petit, exemple :
    1*p^0
num = [1]
# Coefficients au denominateur
# du plus grand ordre au plus petit, exemple :
    1*p^2 + 0.1*p^1 + 1*p^0
# den = [1, 0.1, 1]

plt.subplot(2, 1, 1)
for i in range(5):
    den = [1,0.1+i/5,1]
    #definition de la fonction de transfert
    s1 = signal.lti(num, den)
    # Specification de la plage de pulsations :
    # 0.1 a 10 non inclus par pas de 0.02
    # w, gain, phase : listes de nombres (
        pulsations, gains, phases)
    w, gain, phase = signal.bode(s1, np.arange
        (0.1, 10, 0.02))
    # Trace du graphe en semilog
    plt.semilogx (w, gain, color="blue",
        linewidth="1")
plt.xlabel ("Pulsation␣$\omega$")
plt.ylabel (r"Gain␣$G_{dB}(\omega)=20␣\times␣\
    log_{10}(|H(j\omega)|)$",size=16)
plt.xscale('log')
plt.grid(True,which="both")


plt.subplot(2, 1, 2)
for i in range(5):
    den = [1,0.1+i/5,1]
    s1 = signal.lti(num, den)
    w, gain, phase = signal.bode(s1, np.arange
        (0.1, 10, 0.02))
    plt.semilogx (w, phase, color="red",
        linewidth="1.1")
plt.xlabel ("Pulsation␣$\omega$")
plt.ylabel (r"Phase␣$\phi(\omega)=arg(H(j\omega)
    )$",size=16)
plt.xscale('log')
plt.grid(True,which="both")

plt.show()
```

Annexe 2 : Fonctions Python

——— types ———

str(x) : Return a string version of the x object. If object is not provided, returns the empty string.

int(x, base=10) : Convert a number or string x to an integer, or return 0 if no arguments are given. If x is a number, return x.__int__(). For floating point numbers, this truncates towards zero.

If x is not a number or if base is given, then x must be a string, bytes, or bytearray instance representing an integer literal in radix base.

float(x) : Return a floating point number constructed from a number or string x.

If the argument is a string, it should contain a decimal number, optionally preceded by a sign, and optionally embedded in whitespace. The optional sign may be '+' or '-'; a '+' sign has no effect on the value produced.

——— fichiers ———

open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None) : Open file and return a corresponding file object. If the file cannot be opened, an OSError is raised.

file is either a string or bytes object giving the pathname.

mode is an optional string that specifies the mode in which the file is opened. It defaults to 'r' which means open for reading in text mode. Other common values are 'w' for writing (truncating the file if it already exists), 'x' for exclusive creation and 'a' for appending (which on some Unix systems, means that all writes append to the end of the file regardless of the current seek position).

close() : Flush and close this stream. This method has no effect if the file is already closed. Once the file is closed, any operation on the file (e.g. reading or writing) will raise a ValueError. As a convenience, it is allowed to call this method more than once; only the first call, however, will have an effect.

readline(size=-1) : Read until newline or EOF and return a single str. If the stream is already at EOF, an empty string is returned. If size is specified, at most size characters will be read.

write(s) : Write the string s to the stream and return the number of characters written.

——— chaînes de caractères ———

split(str=" ") : Method that returns a list of all the words in the string, using str as the separator (splits on all whitespace if left unspecified), optionally limiting the number of splits to num.

——— bibliothèque numpy ———

Fonctions mathématiques : **log(x)**, **exp(x)**, **cos(x)**, **sin(x)**, etc.

L'étude proposée porte sur la réplique terrestre [1] du système InSIGHT (**In**terior exploration using **S**eismic **I**nvestigations, **G**eodesy and **H**eat **T**ransport), projet du CNES (Centre National d'Études Spatiales) qui a pour but de déployer une station d'étude de la structure interne de la planète Mars.

Objectif Ecrire un programme permettant de gérer les signaux de commandes des pieds du SEIS pour le maintenir en position à partir de mesures réalisées par des capteurs de position à ultrasons.

On dispose d'une carte Arduino Uno qui peut être programmée dans différents langages. **On se limitera à l'utilisation du langage de programmation Python pour l'étude proposée**.

Le calculateur (la carte Arduino dans notre cas), qui contrôle le mouvement des trois vérins électriques, génère, pour chaque vérin, un signal de consigne rapide (vitesse maximale du moteur) ou lent (un dixième de la vitesse maximale du moteur) en fonction de l'avance de celui-ci. Dans une première phase, il génère une consigne dite « rapide » jusqu'à atteindre une distance de 10 mm entre la tige du vérin et le sol. Le système de commande délivre alors une consigne dite « lente » afin de limiter les contraintes lors du contact entre chaque vérin et le sol. Lors de cette deuxième phase (consigne « lente »), un asservissement en position de chaque vérin permet de maintenir le châssis du SEIS en position horizontale par rapport au sol.

On note pour la suite de l'étude (**figure ??**) :

- `distance` : la variable, de type float, correspondant à la distance mesurée entre le sol et le capteur donnée en cm ;
- `distance_verin` : la variable, de type float, correspondant à la distance entre le capteur et l'extrémité de la tige du vérin donnée en cm ;
- `rapide` et `lente` : variable globale avec des valeurs prédéfinies (correspond aux consignes de la commande du vérin électrique : vitesse rapide, vitesse lente).

Question 13 *Écrire une fonction* `consigne(distance,distance_` *qui calcule l'écart entre la tige du vérin et le sol et qui retourne la consigne* `rapide` *si cet écart est supérieur à 10 mm ou la consigne* `lente` *sinon.*

Le module SEIS est équipé de trois capteurs de positions, à ultrasons, associés à chaque vérin électrique. Chaque capteur est constitué d'un émetteur et d'un récepteur à ultrasons. Le principe de mesure des capteurs

1. Utilisée sur Terre pour validation des différents sous-systèmes.

à ultrasons utilisés est donné ci-dessous et illustré sur la **figure ??**.

Pour déclencher une mesure, il faut présenter une impulsion « High » (5 V) d'au moins 10 μs sur l'entrée « Trig » du capteur (sortie de la carte Arduino). L'émetteur à ultrasons délivre alors une série de 8 impulsions ultrasoniques à 40 kHz, puis il attend le signal réfléchi. Lorsque celui-ci est détecté par le récepteur à ultrasons, le capteur impose un signal « High » sur la sortie « Echo » (entrée de la carte Arduino) dont la durée, tc, est proportionnelle à la distance mesurée. La distance est obtenue en

multipliant la durée du signal tc en seconde par le coefficient constant 17 150 pour obtenir la valeur de la distance en cm.

3 Tableaux

Dans le cas d'un carré magique normal et d'une valeur de n impaire, il existe une méthode simple de construction.

1. Nous notons x et y les numéros de colonne et de ligne $(x,y) \in [0,1,\cdots,n-1]^2$.
2. Dans tous les carrés impairs, il y a une case centrale située de coordonnée $((n-1)/2,(n-1)/2)$, on commence par remplir avec le chiffre 1, la cellule juste à gauche de cette cellule centrale. 1.2
3. On continue ensuite à remplir les autres cases avec la suite des entiers jusqu'à n^2, en suivant les règles suivantes, à partir des coordonnées (x,y) :
   - si le chiffre que l'on vient de placer était un multiple de n on place le nouveau chiffre en $(x-2,y)$ **modulo** n ;
   - sinon on place le nouveau chiffre à la case de coordonnées $(x-1,y-1)$ **modulo** n.

On prendra soin de représenter ce carré magique, qui est une matrice, à l'aide d'une liste de listes. Par exemple, pour le carré magique de taille 3 suivant on utilisera le code suivant :

Code : `A=[[2,7,6],[9,5,1],[4,3,8]]` et **Résultat :** $A = \begin{pmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{pmatrix}$

Question 14 *Continuez de compléter la carré magique de la table* **??** *en utilisant la méthode proposée. Testez les 3 propriétés du carré magique sur cet exemple.*

Question 15 *Proposer une fonction* `def Carre_vide(n:int) -> list :` *qui crée et renvoie un carré magique vide (n listes remplies de n 0), et qui renvoie une liste vide si n est pair.*

Question 16 *Proposer une fonction* `def Remplir_carre(CarreVide : list) -> None :` *qui complète et renvoie un carré magique à partir d'un carré vide.*

Question 17 *Proposer une fonction* `def Verif_carre(Carre : list) -> bool :` *qui vérifie les trois propriétés d'un carré magique et renvoie un booléen (*`True` *ou* `False`*)*

Exercice 94 –

1.1 Indication sur les méthodes associées aux chaînes de caractères

Les attributs suivants s'appliquent à des variables de type de chaîne de caractère :

- `.isalpha` renvoie `True` si c'est une des 26 lettres de l'alphabet et `False` sinon.

```
>>> 'c'.isalpha()
        True
>>> '1'.isalpha()
        False
```

1.3

- `.index(x)` renvoie l'indice de la première occurrence de `x` :

```
>>> 'hello'.index('l')
        2
```

- `.count(x)` renvoie le nombre d'occurrences de `x` :

```
>>> 'hello'.count('l')
        2
```

Le chiffrement de César

Le **chiffrement de César** est un des tout premier code de chiffrement qui ait existé.

La méthode est simple : il suffit de décaler toutes les lettres de l'alphabet du même nombre de lettres.

Par exemple en choisissant un décalage de 3, le A devient le D, le B devient le E, le C devient le F et ainsi de suite. Pour la fin de l'alphabet, il suffit de revenir au début : le W devient Z, le X devient A, le Y devient B et le Z devient C. Ainsi un message comme « la metamorphose » devient par décalage d'une lettre « mb nfubnpsqiptf », ce qui est incompréhensible pour le non initié.

Question 1 *Avec des instructions python, définir trois chaînes de caractères nommées alphabet, mess et code contenant respectivement les caractères de l'alphabet, un message à chiffrer (exemple « la metamorphose ») et le message crypté (vide initialement). On se limitera à des lettres minuscules non accentuées. Les autres caractères (espaces, chiffres...) seront gardés tels quels (non chiffrés).*

Question 2 *Comment est codé le message « franz » avec une valeur de décalage $n = 3$?*

Question 3 *Écrire une fonction* `def decalage(c:str, n:int) -> str:` *permettant de renvoyer un caractère chiffré avec le chiffrement de César.*

Question 4 *Établir un algorithme permettant de chiffrer un message par le code de César, pour un décalage n donné. La fonction associée à cet algorithme aura la signature suivante :* `def chiffrement_cesar(mess:str, n:int) -> str:`.

Question 5 *Proposer ensuite l'algorithme de déchiffrement qui affiche le message chiffré en clair, en supposant que n est inconnu. L'utilisateur choisira parmi les déchiffrements proposés celui qui a du sens! La fonction aura la signature suivante :* `def decryptage_cesar(code:str) -> None:`.

Question 6 *Quelle est la faiblesse de ce type de code ? Proposer en deux lignes une méthode permettant de déterminer (casser) la clé.*

Le chiffre de Vigenère

La méthode de chiffrement par lecture de la table est une méthode adaptée « aux humains », ainsi on réfléchira à

l'implémentation d'un algorithme plus efficace en s'aidant de l'exemple.

On dispose de la variable `alphabet= "abcdefghijklmnopqrstuvwxyz"`

On donne le bloc d'instruction suivant :

```
alphabet= "abcdefghijklmnopqrstuvwxyz"
for i in range(26):
    alphabet = alphabet[1:]+alphabet[:1]
    print (alphabet)
```

Question 7 *Combien de lignes seront affichées ? Quelles seront les deux premières lignes affichées ?*

Question 8 *Écrire la fonction* `def generer_table()->list` : *qui retourne la table de Vigenère sous la forme d'une liste de listes.*
Le résultat sera de la forme suivante :

```
[['a','b','c','d','e','f','g','h','i','j','k','l
    ','m','n','o','p','q','r','s','t','u','v','w
    ','x','y','z'],
 ['b','c','d','e','f','g','h','i','j','k','l','m'
    ,'n','o','p','q','r','s','t','u','v','w','x
    ','y','z','a'],
 ... ]
```

On donne deux fonctions permettant de constuire la ligne « clé » de la table **??** à partir d'un mot et d'une clé.

```
1. def generer_cle_1(mot,cle):
2. nb = len(mot)//len(cle)+1
3. ch_cle=nb*cle
4. ch_cle = ch_cle[0:len(mot)]
5. tab_cle = [car for car in ch_cle]
6. return tab_cle
```

```
1. def generer_cle_2(mot,cle):
2. tab_cle = []
3. for i in range(len(mot)):
4. id = i%len(cle)
5. tab_cle.append(cle[id])
6. return tab_cle
```

Question 9 *En 2 à 3 lignes, expliquer les différences entre les 2 fonctions. Commentez les lignes 2 à 5 des deux fonctions.*

Question 10 *Écrire une fonction étant spécifier ainsi :* `code_vigenere(ch:str, cle:str) -> str` *où la chaîne renvoyée correspond à la chaîne codée avec la clé* `cle`. *Chaque ligne sera commentée. Vous pourrez utiliser les fonctions définies précédemment.*

Question 11 *Quel est selon vous l'intérêt de ce codage par rapport à l'algorithme de César ?*

Exercice 95 –

La percolation[2] désigne le passage d'un fluide à travers un solide poreux. Ce terme fait bien entendu référence au café produit par le passage de l'eau à travers une poudre de café comprimée, mais dans un sens plus large peut aussi bien s'appliquer à l'infiltration des eaux de pluie jusqu'aux nappes phréatiques ou encore à la propagation des feux de forêt par contact entre les feuillages des arbres voisins.

L'étude scientifique des modèles de percolation s'est développée à partir du milieu du XXe siècle et touche aujourd'hui de nombreuses disciplines, allant des mathématiques à l'économie en passant par la physique et la géologie.

Choix d'un modèle

Nous allons aborder certains phénomènes propres à la percolation par l'intermédiaire d'un modèle très simple : une grille carrée $n \times n$, chaque case pouvant être ouverte (avec une probabilité p) ou fermée (avec une probabilité $1-p$). La question à laquelle nous allons essayer de répondre est la suivante : est-il possible de joindre le haut et le bas de la grille par une succession de cases ouvertes adjacentes ?

On conçoit aisément que la réussite ou non de la percolation dépend beaucoup de p : plus celle-ci est grande, plus les chances de réussite sont importantes. Nous auront l'occasion d'observer l'existence pour de grandes valeurs de n d'un seuil critique p_0 au delà duquel la percolation a toutes les chances de réussir et en dessous duquel la percolation échoue presque à chaque fois.

Création et visualisation de la grille

Préparation de la grille

Les deux modules essentiels dont nous aurons besoin sont les modules `numpy` (manipulation de tableaux bi-dimensionnels) et `mathplotlib.pyplot` (graphisme), qu'il convient d'importer :

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

Nous aurons aussi besoin de la fonction `rand` du module `numpy.random` (fonction qui retourne un nombre pseudo- aléatoire de l'intervalle $[0,1[$ et accessoirement de la fonction `ListedColormap` du module `matplotlib.colors` (pour choisir l'échelle chromatique à utiliser pour la représentation graphique). Ces deux fonctions seront importées directement, puisque nous n'aurons pas besoin des modules entiers :

```
from numpy.random import rand
from matplotlib.colors import ListedColormap
```

La grille de percolation sera représentée par le type `np.array`. La fonction `np.zeros((n, p))` renvoie un tableau de n lignes et p colonnes contenant dans chacune de ses cases le nombre flottant 0.0. Une fois un tableau `tab` créé, la case d'indice (i,j) est référencée indifféremment par `tab[i][j]` ou par `tab[i,j]` et peut être lue et modifiée (comme d'habitude, les indices débutent à 0). Enfin, on notera que si tab est un tableau, l'attribut `tab.shape` retourne le couple (n,p) de ses dimensions verticale et horizontale (le nombre de lignes et de colonnes, `tab` étant vu comme une matrice).

Dans la suite de ce document, on représentera une grille de percolation par un tableau $n \times n$, les cases fermées contenant le nombre flottant 0.0 et les cases ouvertes le nombre flottant 1.0.

2. du latin *percolare* : couler à travers.

Question 1 *Définir une fonction* `Python, creationgrille(p, n)` *à deux paramètres : un nombre réel p (qu'on supposera dans l'intervalle $[0,1[$ et un entier naturel n, qui renvoie un tableau (n,n) dans lequel chaque case sera ouverte avec la probabilité p et fermée sinon.*

Visualisation de la grille

Pour visualiser simplement la grille, nous allons utiliser la fonction `plt.matshow` : appliquée à un tableau, celle-ci présente ce dernier sous forme de cases colorées en fonction de leur valeur.

Les couleurs sont choisies en fonction d'une échelle chromatique que vous pouvez visualiser à l'aide de la fonction plt.colorbar(). Celle utilisée par défaut va du bleu au rouge ; puisque nos grilles ne contiennent pour l'instant que les valeurs 0 ou 1, les cases fermées apparaîtrons en bleu, et les cases ouvertes, en rouge.

Changer l'échelle chromatique

L'argument par défaut `cmap` de la fonction `plt.matshow` permet de modifier l'échelle chromatique utilisée. La fonction `ListedColormap` va nous permettre de créer l'échelle de notre choix. Puisque nous n'aurons que trois états possibles (une case pleine représentée par la valeur 0.0), une case vide représentée par la valeur 1.0 et plus tard une case remplie d'eau représentée par la valeur 0.5) une échelle à trois couleurs suffit. Vous pouvez utiliser celle-ci :

```
echelle = ListedColormap(['black', 'aqua', '
    white'])
```

Question 2 *Écrire une fonction* `afficher_grille(grille,nom_de_fichier)` *qui prend en argument une variable grille qui correspond à une grille de percolation générée précédemment et ne renvoyant rien mais enregistrant dans* `nom_de_fichier` *le graphe obtenu. On pourra exporter une grille de 10×10 cases avec l'échelle suggérée précédemment, l'enregistrer sous le nom* `tp09_q02_vos_noms.png` *et l'envoyer à votre professeur.*

Percolation

Une fois la grille crée, les cases ouvertes de la première ligne sont remplies par un fluide, ce qui sera représenté par la valeur 0.5 dans les cases correspondantes. Le fluide pourra ensuite être diffusé à chacune des cases ouvertes voisines d'une case contenant déjà le fluide jusqu'à remplir toutes les cases ouvertes possibles.

Question 3 *Écrire une fonction* `percolation(grille)` *qui prend en argument une grille et qui remplit de fluide celle-ci, en appliquant l'algorithme exposé ci-dessous :*

1. Créer une liste contenant initialement les coordonnées des cases ouvertes de la première ligne de la grille et remplir ces cases de liquide.
2. Puis, tant que cette liste n'est pas vide, effectuer les opérations suivantes :
   (a) extraire de cette liste les coordonnées d'une case quelconque ;
   (b) ajouter à la liste les coordonnées des cases voisines qui sont encore vides, et les remplir de liquide.

L'algorithme se termine quand la liste est vide.

Question 4 *Rédiger un script vous permettant de visualiser une grille avant et après remplissage, et faire l'expérience avec quelques valeurs de p pour une grille de taille raisonnable (commencer avec $n=10$ pour vérifier visuellement que votre algorithme est correct, puis augmenter la taille de la grille, par exemple avec $n=64$). On pourra exporter et l'enregistrer sous le nom* `tp09_q04_vos_noms.png` *et l'envoyer à votre professeur.*

On dit que la percolation est réussie lorsqu'à la fin du processus au moins une des cases de la dernière ligne est remplie du fluide.

Question 5 *Écrire une fonction* `teste_percolation(p,n)` *qui prend en argument un réel $p\in[0,1[$ et un entier $n\in\mathbb{N}^*$, crée une grille, effectue la percolation et retourne :*

- `True` lorsque la percolation est réussie, c'est-à-dire lorsque le bas de la grille est atteint par le fluide ;
- `False` dans le cas contraire.

Seuil critique

Nous allons désormais travailler avec des grilles de taille 128×128 [3]

Notons $P(p)$ la probabilité pour le fluide de traverser la grille. Pour déterminer une valeur approchée de la probabilité de traverser la grille, on se contente d'effectuer k essais pour une valeur de p puis de renvoyer le nombre moyen de fois où le test de percolation est vérifié.

Question 6 *Rédiger la fonction* `proba(p,k,n)` *qui prend en argument le nombre d'essai k, la variable p ainsi que le nombre de cases n sur la largeur de la grille et qui renvoie $P(p)$.*

Question 7 *Écrire une fonction* `tracer_proba(n,nom_de_fichie` *qui prendre en argument une taille n ne renvoyant rien mais enregistrant dans* `nom_de_fichier` *le graphe obtenu. On pourra traiter le cas d'une grille de 128×128 cases et enregistrer la figure obtenue sous le nom* `tp09_q07_vos_noms.png` *et l'envoyer à votre professeur.*

Exercice 96 –

Écrire une fonction `supprime` prenant deux arguments, un tableau `t` d'entiers et un entier `n`, et renvoyant un tableau dont les éléments sont ceux de `t`, privé des occurrences de `n`.
Par exemple, si `t=[2,1,6,2,8,6,2,1]` et `n=2`, alors le tableau renvoyé sera `[1,6,8,6,1]`.

Exercice 97 –

Étant donnés deux vecteurs `u` et `v` de même taille, de coordonnées $(u_0,u_1,\cdots,u_{n-1})$ et $(v_0,v_1,\cdots,v_{n-1})$, on définit la somme `u+v` de coordonnées $(u_0+v_0,u_1+$

3. Baisser cette valeur si le temps de calcul sur votre ordinateur est trop long.

$v_1, \cdots, u_{n-1}+v_{n-1}$), et le produit scalaire `u.v`, qui est le réel $\sum_{k=0}^{n-1} u_k v_k$.

On choisit de représenter tout vecteur $(x_0, x_1, \cdots, x_{n-1})$ par le tableau $[x_0, x_1, \cdots, x_{n-1}]$. Écrire alors deux fonctions calculant la somme et le produit scalaire de deux vecteurs.

Exercice 98 –

On représente une matrice par un tableau l'élément i est un tableau contenant les coefficients de la $(i+1)$-ème ligne de la matrice.

Par exemple, si $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, M est représentée par le tableau $[[1,2,3],[4,5,6]]$.

Écrire une fonction prenant deux matrices comme argument et retournant leur produit matriciel s'il existe, et un message d'erreur sinon.

Exercice 99 –

Question 1

Écrire une fonction `pascal(n)` ayant comme argument un entier naturel `n` et retournant la `n`$^{\text{e}}$ ligne du triangle de Pascal, sous forme de tableau. Ainsi, pour $n=2$, cette fonction doit retourner $[1,2,1]$.

Attention : seul l'usage de la formule de Pascal est autorisé; en particulier, il est interdit d'utiliser la relation $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Exercice 100 –

Écrire une fonction qui, étant donné un tableau, retourne ce tableau, dans lequel les occurrences du minimum et celles du maximum ont été échangées.

Exercice 101 –

Écrire une fonction qui, étant donné un tableau d'entiers trié (dans l'ordre croissant), retourne l'indice du premier élément du plus grand sous-tableau constant.

Exercice 102 –

Étant donné un tableau $t = [t_0, \ldots, t_{n-1}]$ d'entiers de longueur n, on appelle sous-tableau croissant de t un tableau $[t_{i_0}, \ldots, t_{i_{k-1}}]$ (donc, de longueur k), avec

- $0 \leq i_0 < i_1 < \cdots < i_{k-1} \leq n-1$;
- $t_{i_0} \leq t_{i_1} \leq \cdots \leq t_{i_{k-1}}$.

On considérera qu'un tableau vide, donc de longueur 0, est croissant.

On s'intéresse au problème de la recherche du plus long sous-tableau croissant dans un tableau d'entiers.

Question 1

Quelle est la longueur minimale d'un tel plus long sous-tableau croissant? Quand est-elle atteinte?

Question 2

Que dire si l'on trouve un plus long sous-tableau croissant de longueur n?

Question 3

Écrire une fonction `est_croissant(t)` renvoyant un booléen indiquant si le tableau d'entiers `t` est croissant.

1.4 Recherche exhaustive.

On cherche d'abord à mettre en œuvre une stratégie naïve : on explore tous les sous-tableaux de t (là, vous devriez vous dire que ce n'est pas une bonne idée)!

Pour faire cela, rappelons nous que prendre un sous-tableau de t correspond à prendre une partie de $[\![0,n]\!]$. Or, l'ensemble des parties de $[\![0,n]\!]$ est en bijection avec $\{0,1\}^{[\![0,n[\![}$. Un sous-tableau v de t est donc caractérisé par une famille $(e_0,\ldots,e_{n-1}) \in \{0,1\}^{[\![0,n[\![}$, avec, si $0 \leq i < n$, $e_i = 1$ si et seulement si $t[i]$ est présent dans v. Enfin, on peut voir que pour obtenir toutes les familles de $\{0,1\}^{[\![0,n[\![}$, il suffit d'écrire la décomposition binaire sur n bits de tous les entiers entre 0 et 2^n-1.

▪ **Exemple** Avec `t = [1,5,3,2]`, de longueur 4, le sous-tableau de `t` décrit par `[0,0,0,0]` est `[]`, celui décrit par `[1,0,0,1]` est `[1,2]` et celui décrit par `[0,1,1,1]` est `[5,3,2]`. ▪

On peut alors penser à implémenter l'idée suivante : on parcourt séquentiellement les entiers entre 0 et 2^n-1, on calcule l'écriture binaire de chaque entier, cela définit un sous-tableau de t. Il ne reste plus qu'à savoir si ce sous-tableau est croissant et à calculer sa longueur.

Commencez par recopier le code suivant dans votre script.

```
def binaire(k,n):
    """Renvoie le tableau de n bits écrivant k
       en binaire
    Précondition : 0 <= k <= 2**n -1 """
    L = [0]*n
    p = k
    for i in range(n):
        L[n-1-i] = p % 2
        p = p // 2
    return L
```

Soit `k` un entier écrit en binaire avec `n` chiffres : $\texttt{k} = \underline{a_{\texttt{n}-1}\ldots a_1 a_0}_2$, c'est-à-dire que

$$\texttt{k} = \sum_{j=0}^{\texttt{n}-1} a_j 2^j.$$

Question 4

Écrire un invariant portant sur `L` et `p` dans la boucle `for` de la fonction `binaire(k,n)` et justifier que cette fonction renvoie bien le résultat demandé.

Question 5

Écrire une fonction `longueur(e)` qui prend en argument un tableau `e` contenant des 0 et des 1 et qui renvoie le nombre de 1 dans `e`.

Question 6

Écrire une fonction `extrait(t,e)` qui prend en argument un tableau `t` d'entiers et un tableau `e` de 0 et de 1 et qui renvoie le sous-tableau de `t` désigné par `e`.

Question 7

Écrire une fonction `stc_exhaustif(t)` donnant la longueur du plus grand sous-tableau croissant de `t`.

1.5 Facultatif : programmation dynamique.

Le programme écrit dans la partie précédente n'est pas très efficace (essayez par exemple de le tester sur un tableau de longueur 100). Nous allons adopter une autre stratégie.

Toujours avec $t=[t_0,\dots,t_{n-1}]$ un tableau d'entiers de longueur n, on note $m=[m_0,\dots,m_{n-1}]$ le tableau de longueur n tel que, si $0\leq i<n$, m_i est la taille du plus grand sous-tableau croissant de t dont le dernier élément est t_i.

Question 8
Que vaut m_0 ?

Question 9
Soit $0\leq i<n-1$, supposons que l'on connaisse t et $[m_0,\dots,m_i]$. Comment calculer m_{i+1} ?

Question 10
Si l'on connaît m, comment obtenir la longueur de la plus grande sous-suite croissante de t ?

Question 11
Écrire une fonction `stc_dynamique(t)` donnant la longueur du plus grand sous-tableau croissant de `t`.

Question 12
Quelle est la différence entre les fonctions écrites dans chaque partie, notamment à l'utilisation ?

Exercice 103 –

Écrire une fonction qui, étant donné un tableau d'entiers de longueur au moins égale à 3, retourne l'indice du premier élément d'un sous-tableau de longueur 3 dont la somme des éléments est maximale.

Exercice 104 –

Soit n un entier naturel. On dit que c'est un *2-palindrome* si son écriture en base 2 est la même qu'elle soit écrite de gauche à droite ou de droite à gauche. Plus précisément, si $n=\sum_{i=0}^{l} a_i 2^i$, avec les a_i dans $\{0,1\}$, et $a_k=1$, n est un 2-palindrome si pour tout i dans $\{0,\cdots,k\}$, on a $a_i=a_{k-i}$. Par exemple les entiers 3 (11), 5 (101), 7 (111), 9 (1001) et 15 (1111) sont des 2-palindromes. Écrire un programme qui calcule les 2-palindromes n tels que $n\leq 511$.

Exercice 105 –

Le but de cet exercice est de construire la suite ordonnée des entiers de la forme $2^p3^q5^r$, où p, q et r sont des entiers naturels.
Cette suite commence par :

valeurs	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**10**	**12**	**15**	**16**
indice	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
				↑ i_5		↑ i_3		↑ i_2				

Principe :

Le tableau `t` étant en cours de construction, l'élément suivant du tableau sera à choisir parmi les nombres suivants : `2*t[i_2]`, `3*t[i_3]` et `5*t[i_5]`, où i_2 (respectivement i_3, i_5) est l'indice du plus petit élément du tableau n'ayant pas son double (respectivement triple, quintuple) présent dans le tableau.
Dans l'exemple, $i_2=8$, $i_3=6$ et $i_5=4$, et l'élément suivant est donc à choisir parmi 2*9=18, 3*6=18 et 5*4=20.
Le plus petit élément étant 18, c'est l'élément à rajouter au tableau. Les indices concernés par le choix (ici i_2 et i_3) sont à augmenter de 1, ce qui donne le tableau suivant :

valeurs	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**10**	**12**	**15**	**16**
indice	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
				↑ i_5			↑ i_3		↑ i_2			

1. Écrire, selon ce principe, la fonction qui construit le tableau des n premiers nombres de la forme $2^p3^q5^r$.
2. Donner l'invariant de boucle et la démonstration de ce théorème.

Exercice 106 –

Le but de cet exercice est de trier un tableau `t` dont les éléments ne peuvent prendre qu'un "petit" nombre de valeurs (ici, trois valeurs : 0, 1 ou 2).
Par exemple, à partir du tableau initial :

valeurs	**2**	**0**	**1**	**0**	**2**	**1**	**1**	**0**	**1**	**2**	**2**	**1**	
indice	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*1*

il faut obtenir le tableau final :

valeurs	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**2**	**2**	**2**	
indice	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*1*

Le programme sera itératif (i.e. on utilisera une boucle `for`). Pour trouver comment écrire le corps de la boucle, on suppose que le tableau `t` (de taille n) a été traité jusqu'au rang i, et que sont connus j et k vérifiant :

- tous les éléments du tableau d'indice inférieur ou égal à j sont égaux à 0 ;
- tous les éléments du tableau d'indice supérieur strictement à j et inférieur ou égal à j sont égaux à 1 ;
- tous les éléments du tableau d'indice supérieur strictement à k sont égaux à 2.

0	⋯	0	1	⋯	1	2	⋯	2	x	⋯
1		j	$j+1$		k	$k+1$		i	$i+1$	n

Le prochain élément du tableau à traiter est `t[i+1]`, noté `x`.
• En supposant $1\leq j<k<i<n$ (c'est-à-dire qu'il y a au moins un 0, un 1 et un 2 placés), quelles sont les modifications à faire sur `t`, i et j si $x=2$? si $x=1$? si $x=0$?

Il se peut qu'il n'y ait pas encore de 0, 1 ou 2 dans la partie de tableau déjà triée. Il faut en tenir compte dans les modifications de `t`, i et j selon les valeurs de x.

• Écrire le traitement complet du tri.

Exercice 107 –

On définit la fonction de Kaprekar comme suit (on la note K). Si $n\in\mathbb{N}$, on écrit n en base 10 (sans zéros inutiles), on prend c le nombre obtenu en écrivant les chiffres de n dans l'ordre croissant et d celui obtenu en écrivant les chiffres de n dans l'ordre décroissant. On pose alors $K(n)=d-c$.

▪ **Exemple** $K(6384)=8643-3468=5175$, $K(5175)=5994$, $K(5355)=5994$, $K(5994)=5355$, $K(5355)=1998$, $K(1998)=8082$, $K(8082)=8532$, $K(8532)=6174$ et $K(6174)=6174$. ▪

Toutes les suites récurrentes ainsi construites bouclent. Ici,

nous avons un cas particulier : la suite est stationnaire.

Question 1 *C*

onstruire une fonction `Kaprekar(n)` prenant en argument un entier `n` et donnant en sortie les valeurs de la suite décrite plus haut, jusqu'à ce qu'elle ne boucle.

▪ **Exemple** L'appel de `Kaprekar(6384)` devra donner comme résultat
`[6384, 5175, 5994, 5355, 1998, 8082, 8532, 6174]`. ▪

Indice : on pourra écrire une fonction convertissant un entier en la liste de ses chiffres en base 10, une fonction convertissant une liste de chiffres en entier, une fonction calculant K et utiliser la méthode `sort`.

Exercice 108 –

Dans un tableau t contenant $2N$ ou $2N+1$ nombres, une médiane est un réel m tel que, si l'on classe t par ordre croissant,

1. m est supérieur ou égal aux N premiers nombres de t ;
2. m est inférieur ou égal aux N derniers nombres de t.

▪ **Exemple**
- La seule médiane de $[1,3,2]$ est 2.
- La seule médiane de $[5,1,2,5,5]$ est 5.
- La seule médiane de $[0,1,0,0]$ est 0.
- L'ensemble des médianes de $[-3,4,1,-1]$ est l'intervalle $]-1,1[$.

▪

Question 1 ´

Ecrire une fonction `mediane(t)` qui, à un tableau de nombres `t`, renvoie une médiane de `t`.

On pourra s'inspirer de l'une des deux idées suivantes.
- *Copier* `t`*, lui ôter tant que possible son maximum et son minimum.*
- *Trier* `t` *par ordre croissant.*

Exercice 109 –

Question 1 ´

Ecrire une fonction `miroir(t)` qui, à un tableau `t` de nombres, renvoie le tableau contenant les même nombres, renversé.

▪ **Exemple** Avec `t = [5,2,6,3,7]`, `miroir(t)` renverra `[7,3,6,2,5]`. ▪

Exercice 110 –

Question 1 ´

Ecrire une fonction `doublon(t)` qui, à un tableau `t`, renvoie un booléen indiquant s'il existe un élément de `t` se trouvant au moins deux fois dans `t`.

Exercice 111 –

Dans un tableau $t = [t_0, \dots, t_{n-1}]$, on dit que l'on a un record à une position $0 \leq k < n$ si $\forall i < k,\ t_i < t_k$. Par définition, on a toujours un record en position 0.

Question 1

Écrire une fonction `record(t,k)` qui, à un tableau de nombres `t` et un entier `k`, renvoie le booléen `True` si `t` admet un record en position `k`, `False` sinon.

Question 2

Écrire une fonction `nb_records(t)` qui, à un tableau de nombres `t`, renvoie le nombre de records dans `t`.

Exercice 112 –

On rappelle que, si un entier naturel $p \geq 2$ n'a aucun diviseur inférieur à $\sqrt{p}$ (sauf 1), alors p est premier.

Question 1

Écrire une fonction `crible(p)` qui, à un entier naturel `p`, renvoie le tableau des nombres premiers inférieurs ou égaux à p^2, triés par ordre croissant.

Exercice 113 –

Question 1

Écrire une fonction `Python` renvoyant un indice du maximum d'un tableau de nombres `T`.

Question 2

Écrire un invariant de boucle pour cet algorithme. Démontrer que l'algorithme précédent donne le bon résultat.

Question 3

Donner la complexité dans le pire des cas du calcul d'un indice du maximum de `T` par cette fonction, en fonction de la longueur de `T`, que l'on notera n.

Exercice 114 –

Soit `t` $= [t_0, \dots, t_{n-1}]$ un tableau de nombres de longueur $n \in \mathbb{N}^*$, soit $k \in \mathbb{N}^*$ et $i \in [\![0, n+2-2k[\![$.

On dit que `t` possède une *pyramide* de *hauteur* k en position i si

$$t_i < t_{i+1} < \dots t_{i+k-1}$$

et si

$$t_{i+k-1} > t_{i+k} > \dots t_{i+2k-2}.$$

Par exemple, il y a une pyramide de longueur 1 en tout élément de `t`, et il y a une pyramide de longueur 2 en position i si

$$t_i < t_{i+1} \quad \text{et} \quad t_{i+1} > t_{i+2}.$$

Ainsi, avec

$$\texttt{t} = [-1,0,4,2,-3,0,5,1],$$

`t` a une pyramide de hauteur 3 en position 0, des pyramides de hauteur 2 en position 1 et 5 et des pyramides de hauteur 1 en toute position.

Question 1

Écrire une fonction `nb_pyramides_2(t)` renvoyant le nombre de pyramides de hauteur 2 présentes dans le tableau de nombres `t` passé en argument.

Question 2

Écrire une fonction `pyramide_a_partir(t,i)` prenant en argument un tableau de nombres `t` et un indice `i` et renvoyant la taille de la pyramide du tableau `t` débutant en position `i`.

Question 3

Écrire une fonction `plus_haute_pyramide(t)` renvoyant la taille de la plus haute pyramide présente dans le tableau de nombres `t` passé en argument. On pourra utiliser la fonction `pyramide_a_partir(t,i)` de la question précédente, même si cette question n'a pas été résolue.

Exercice 115 –

Soit `t` $= [t_0, \ldots, t_{n-1}]$ un tableau de nombres de longueur $n \in \mathbb{N}^*$, soit $0 \leq i, j \leq n-1$.

On dit qu'il y a un *ascension* dans le tableau `t` aux indices i et j si

$$i < j \quad \text{et} \quad t_i < t_j.$$

La *hauteur* de cette ascension est alors $t_j - t_i$.

On propose la fonction suivante.

```
"""Plus haute ascension de t, ou 0 s'il n'y
    en a pas.
Précondition : t est un tableau de nombres
    """
M = 0
n = len(t)
for i in range(n):
    m = 0
    for j in range(i+1,n):
        if t[j] - t[i] > m :
            m = t[j] - t[i]
    if m > M :
        M = m
return M
```

Question 1

Montrer que « si `m` > 0, alors `m` est la hauteur de la plus haute ascension de `t[i:j]`, sinon $m = 0$ » est un invariant de boucle pour la boucle `for` portant des lignes 8 à 10 de la fonction `plus_haute_ascension(t)`.

Question 2

Justifier qu'un appel de la fonction `plus_haute_ascension(t)` renvoie bien le résultat demandé, à l'aide notamment d'un invariant.

Question 3

Écrire une fonction `nb_ascensions(t)` renvoyant le nombre d'ascensions présentes dans le tableau de nombres `t` passé en argument.

Exercice 116 –

On considère la fonction suivante.

```
"""Précondition : n entier naturel"""
u = 2
for i in range(n) :
    u = 4 * u - 3
return u
```

Question 1

Montrer que « $u = 4^i + 1$ » est un invariant d'entrée de boucle, pour la boucle for de la fonction `suite(n)`. En déduire le résultat renvoyé par cette fonction.

Exercice 117 –

On considère la fonction suivante.

```
"""Préconditions : L tableau de nombres, M
    nombre"""
S = 0
n = len(L)
for i in range(n) :
    if L[i] > M :
        S = S + L[i]
return S
```

Question 1

Que renvoie un appel de la fonction `mystere(L,M)` ? On justifiera la réponse, à l'aide notamment d'un invariant.

Exercice 118 –

On considère la fonction suivante.

```
"""Préconditions : n entier positif"""
L = []
c = 0
while c ** 2 <= n :
    L.append(c**2)
    c = c+1
return L
```

Question 1

Montrer que, si `n` est un entier, un appel de la fonction `mystere(n)` renvoie un résultat, à l'aide d'un variant.

Question 2

Que renvoie un appel de la fonction `mystere(n)` ? On justifiera la réponse, à l'aide notamment d'un invariant.

Exercice 119 –

Dans un tableau de nombres $t = [t_0, \ldots, t_{n-1}]$ de longueur n, la position $1 \leq i < n-1$ est dit *sous l'eau* s'il existe $k \in [\![0, i[\![$ et $\ell \in [\![i+1, n[\![$ tels que

$$t_k > t_i \quad \text{et} \quad t_\ell > t_i.$$

Question 1

Écrire une fonction `nb_sousleau(t)` prenant en argument un tableau de nombres t et renvoyant le nombres d'indices sous l'eau de ce tableau.

Exercice 120 –

Question 1 *Écrire une fonction* `maxi_double(M)` *prenant en argument une matrice de nombres* `M` *(représentée comme liste de listes) et renvoyant le maximum de* `M` *?*

Question 2 *Écrire une fonction* `liste_imax(t)` *renvoyant la liste des indices où le maximum de* `t` *est atteint.*

Exercice 121 –

Question 1 *Écrire une fonction* `indice(x, t)` *renvoyant un indice* `i` *tel que* `t[i]==x` *si* `x` *apparaît dans le tableau* `t` *et* -1 *sinon.*

Question 2 *Écrire une fonction* `tous_les_indices(e,t)` *renvoyant la liste de tous les indices des occurrences de* `e` *dans le tableau* `t`.

Question 3 *Écrire une fonction* `compte(e,t)` *renvoyant le nombre d'occurrences de* `e` *dans le tableau* `t`.

Question 4 *Écrire une fonction* `ind_appartient_dicho(e,t)` *renvoyant l'indice d'une occurrence de* `e` *dans le tableau* `t` *(*`None` *si* `e` *n'est pas dans* `t`*), en supposant que* `t` *est trié par ordre croissant.*

Question 5 *Écrire une fonction* `dec_appartient_dicho(e,t)` *renvoyant un booléen indiquant si* `e` *est dans le tableau* `t`*, en supposant que* `t` *est trié par ordre décroissant.*

Question 6 *Écrire une fonction* `compte(e,t)` *comptant le nombre d'occurences de l'élément* `e` *dans le tableau*

`t`.

Exercice 122 –

- Ce devoir est à faire de façon individuelle.
- Utilisez Python, version 3.
- Créez un répertoire pour faire ce DS.
- Allez sur le site de la classe dans la rubrique Info/DS Tronc Commun.
- Recopiez le fichier `ds.py` dans ce répertoire, ainsi que le fichier `zeta5.txt`. Vous aurez besoin de ces fichiers pour le DS mais vous ne devrez pas modifier que le fichier `ds.py` sans modifier les instructions déjà données (sous peine de risquer d'avoir tout faux). 1.6
- Avec votre IDE (Pyzo ou IDLE) ouvrir le script `ds.py`.
- On ne vous demande pas de rendre votre programme mais seulement de répondre aux questions posées, dont les réponses sont toutes numériques, **sur le formulaire de réponse joint**.
- **Attention : toutes les questions posées dépendent d'un paramètre α, qui vous est donné sur le formulaire de réponse. La valeur de α est différente pour chacun d'entre vous. Soyez attentif à sa valeur : s'il est faux toutes vos réponses seront fausses.** Dans toute la suite, on considère que la variable Python `alpha` contient la valeur de votre paramètre α. Il vous est donc conseillé, au début de `ds.py`, de faire un `alpha = `α.
- Lorsque la réponse demandée est un réel, on attend que l'écart entre la réponse que vous donnez et la vraie valeur soit strictement inférieur à 10^{-4}. Donnez donc des valeurs avec 5 chiffres après la virgule.

Analyse de tableaux

Première partie

Après avoir exécuter votre script `ds.py` (contenant la définition de la variable `alpha` avec votre valeur de α) exécutez `t = cree_tableau(alpha)`.

Question 1 *Quelle est la longueur du tableau* `t` *? Par la suite, on la note N.*

Question 2 *Combien d'éléments de* `t` *sont supérieurs ou égaux à* 3000 *?*

Question 3 *Combien d'éléments de ce tableau sont divisibles par* 3 *?*

Question 4 *Quel est le nombre de couples (i,j) tels que $0 \leq i < j < N$ et* `t[`i`]` $<$ `t[`j`]` *?*

On définit la suite u par

$$\forall n \in \mathbb{N}\ u_{n+1} = r(15091 \cdot u_n, 64007)$$
$$\text{et } u_0 = 10 + \alpha$$

où $r(a,d)$ désigne le reste de la division euclidienne de a par d (en python, on utilise l'opérateur `%` pour calculer ce reste).

Construire un tableau `U` de taille 10^4 tel que pour tout $i \in [\![0, 10^4 - 1[\![$, `U[`$i$`]` contienne u_i.

Question 5 *Que vaut u_{42} ?*

Question 6 *Que vaut le dernier élément du tableau* `U` *?*

Question 7 *Quel est le nombre d'indices $i \in [\![0, 10^4 - 2[\![$ tels que $|u_i - u_{i+1}| \leq 1000$?*

Question 8 *Quelle est la somme des valeurs de ce tableau ?*

Autour de $\zeta(5)$

Dans cette partie, on utilise le fichier `zeta5.txt`. Celui-ci contient un million de décimales (après la virgule) de $\zeta(5) = \sum_{n=1}^{+\infty} \frac{1}{n^5} \approx 1,0369$.

Voici les premières lignes du fichier (à gauche, on a noté en petit les numéros des lignes pour pouvoir en parler; attention : **ils n'apparaissent pas dans le fichier**) :

```
zeta5_huvent = 1.
0369277551 4336992633 1365486457 0341680570
    8091950191 : 50
2811974192 6779038035 8978628148 4560043106
    5571333363 : 100
7962034146 6556609042 8009617791 5597084183
    5110721800 : 150
8764486628 6337180353 5983639623 6512888898
    1335276775 : 200
2398275032 0224368457 6644466595 8115993917
    9777450392 : 250
4464391966 6615966401 6205325205 0215192267
    1351256785 : 300
9748692860 1974479843 2006726812 9753091990
    0774656558 : 350
6015265737 3003756153 2683149897 9719350398
    3785813199 : 400
2288488642 5335104251 6025108499 0434640294
    1172432757 : 450
6341508162 3322456186 4992714427 2264614113
    0075808683 : 500

1691649791 8137769672 5145590158 0353093836
    2260020230 : 550
4558560981 5265536062 6530883832 6130378691
    7412255256 : 600
0507375081 3917876046 9541867836 6657122379
    6259477937 : 650
8931344280 5560465115 0585291073 6964334642
    8934143397 : 700
5231743713 3962434331 1485731093 6262213535
    7253048207 : 750
```

À partir de la ligne 3, les décimales sont rangées par paquets de 10, eux-mêmes rangés par lignes de 5 paquets, elles-mêmes rangées dans des blocs de 10 lignes (qui contiennent donc 500 décimales; il y a ainsi 2000 blocs). Attention : il y a des lignes vides (la ligne 13 sur cet extrait puis de manière générale, tous les paquets de 500 décimales). On va faire des statistiques sur le nombre d'apparitions de $2000+\alpha$ dans l'écriture décimale de $\zeta(5)$. Par exemple, 92 apparaît deux fois dans les 20 premières décimales. Plus subtil : 70 apparaît deux fois dans les 40 premières décimales, dont une fois tronqué par un espace. On peut même trouver des occurrences coincées entre deux lignes (1004,

Modéliser

en position 6199), voire entre deux blocs (1039, en position 15498).

Au besoin, vous pouvez ouvrir ce fichier avec n'importe quel éditeur de texte.

Question 9 *Donner le nombre d'occurrences de* $2000+\alpha$ *dans les paquets de* 10 *décimales. On pourra appliquer un algorithme du type : « Pour chaque ligne l, on casse l selon les espaces. S'il y a le bon nombre de blocs, alors pour chacun des blocs b, et pour chaque i entre* 0 *et* 6 *(inclus), on regarde si $b[i:i+4]$ vaut la chaîne représentant* $2000+\alpha$ *».*

Question 10 *Donner le nombre d'occurrences de* $2000+\alpha$ *dans les paquets de* 50 *décimales obtenus en concaténant, pour chaque ligne, les* 5 *paquets de* 10 *décimales.*

Question 11 *Donner le nombre d'occurrences de* $2000+\alpha$ *dans les paquets de* 500 *décimales obtenus en concaténant, pour chaque bloc de* 10 *lignes, les* 50 *paquets de* 10 *décimales.*

Question 12 *Donner le nombre d'occurrences de* $2000+\alpha$ *dans les* 10^6 *premières décimales de* $\zeta(5)$.

Exercice 123 –

Images en niveaux de gris

On cherche à représenter des images. Pour simplifier, on ne cherchera pas à représenter les couleurs mais seulement la luminosité des différents points de l'image. Pour cela, on découpe l'image, supposée rectangulaire, en carrés de même taille (appelés *pixels*). On remplace alors chaque carré par un entier naturel indiquant la luminosité (moyenne) de l'image sur le carré considéré (la luminosité est donnée dans une unité arbitraire, les valeurs allant de 0 pour un carré noir à N pour un carré blanc, où N est une valeur arbitrairement fixée). On obtient ainsi une matrice de $n \times p$ valeurs entières appartenant à l'intervalle $[0, N]$.

Pour lire et écrire une telle image dans un fichier, on peut utiliser le format de données PGM (Portable GrayMap), dans sa version texte (*plain format*).

Ce format de données consiste à représenter une image de la façon suivante par un fichier.

- Le fichier est un fichier texte (ne comportant que des caractères ASCII).
- On appellera *blanc* tout caractère qui est soit un retour à la ligne, soit un caractère espace, soit un caractère de tabulation (en fait, un autre caractère)
- Toutes les valeurs contenues dans le fichier sont séparées par un ou plusieurs blancs. [4]
- La première ligne du fichier doit contenir la «valeur magique» constituée des deux caractères «P2» (cette contrainte sert à distinguer un fichier pgm en niveaux de gris d'un autre type de fichier) et doit être suivie d'un blanc.
- Les autres valeurs écrites dans le fichier sont toutes des entiers naturels, écrits en base 10 (autrement dit, «douze» est représenté par la succession des caractères 1 et 2).
- Après la valeur magique, on trouve un nombre représentant la largeur p de la matrice puis un nombre représentant la hauteur n de l'image, puis un nombre donnant la valeur de N choisie pour cette image (intensité de gris représentant le blanc), puis toutes les valeurs de la matrice représentant l'image, dans l'ordre où on les lirait normalement si la matrice était un texte écrit en français (c'est-à-dire de gauche à droite puis de haut en bas).
- Le fichier peut contenir des commentaires ; ceux-ci commencent par le caractère `#` et finissent avec le retour à la ligne suivant. Ils doivent être simplement ignorés. Pour faciliter le travail demandé par la suite, *on supposera qu'il n'y a aucun commentaire dans les fichiers que l'on traitera.*
- On doit avoir $N \in [\![0, 2^{16}[\![$.
- Les lignes du fichier doivent faire au plus 70 caractères.

On rappelle qu'on peut convertir une chaîne représentant un nombre décimal avec `int` et réciproquement, convertir un nombre en chaîne le représentant avec `str`.

Enfin, on représentera une matrice (de nombres) à n lignes et p colonnes comme un tableau (ou liste Python) de longueur n, chacun de ses éléments étant un tableau de longueur p. On décrit donc la matrice ligne par ligne. Ainsi, l'indice « ligne `i` colonne `j` » de la matrice représentée par `M` sera `M[i][j]`.

▪ **Exemple** La matrice $\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix}$ sera représentée par `[ [0,1,2] , [3,4,5] ]`. ▪

Travail demandé

Commencez par recopier les fonctions `image_noire`, `dim` et `lit_valeurs` données ci dessous : ce sont des exemples utiles, utilisez les !

```
def image\_noire(n, p):
    """Construit la matrice n*p d'une image
       noire."""
    img = [0]*n
    for i in range(n):
        img[i] = [0]*p
    return img

def dim(img):
   """Donne le couple (n, p) des dimensions de
      la matrice img. n :
   nombre de lignes, p : nombre de colonnes. La
      matrice est supposée
   avoir au moins une ligne."""
   n = len(img)
   p = len(img[0])
   return (n,p)

def lit\_valeurs(nom\_de\_fichier):
    """Lit le contenu du fichier image f et
       renvoie la liste des
```

4. À toutes fins utiles, pour toute chaîne de caractères `s`, l'expression `s.split()` désigne la liste obtenue par découpage de `s` en utilisant les blancs comme séparateurs.

```
    valeurs lues (séparées par des blancs)
        sous forme d'une liste
    de chaines de caractères. La première
        valeur est normalement
    'P2'."""
    with open(nom\_de\_fichier, 'r') as f:
        c = f.read()
    return c.split()
```

N'hésitez pas à ouvrir l'image `degrade.pgm` et `joconde.pgm` dans un éditeur de texte puis dans un lecteur d'images, afin de comprendre le codage des images.

Sauvegarde d'images

Question 1

Écrire une fonction `sauve_image(img, N, nom_de_fichier)` prenant en argument une matrice `img` représentant une image, l'entier `N` comme niveau de gris maximal (dans $[\![0, 2^{16}[\![$) ainsi qu'une chaîne `nom_de_fichier` et sauvant l'image dans le fichier nommé `nom_de_fichier`, au format PGM.

Question 2

Écrire une fonction `sauve_rectangle_noir(n, p, N, nom_de_fichier)` sauvant dans le fichier `nom_de_fichier` un rectangle noir, de côté n pixels de hauteur et p de largeur, où le blanc est d'intensité N. Vérifier que l'image produite par

```
sauve\_rectangle\_noir(100, 200, 255, '
    rectangle\_noir.pgm')
```

est bien ce que vous attendiez grâce à GIMP ou la visionneuse d'images.

Question 3

Écrire une fonction `sauve_rectangle_blanc(n, p, N, nom_de_fichier)` sauvant dans le fichier `nom_de_fichier` un rectangle blanc, de côté n pixels de hauteur et p de largeur, où le blanc est d'intensité N. De même, vérifiez l'image produite par

```
sauve\_rectangle\_blanc(100, 200, 255, '
    rectangle\_blanc.pgm').
```

Question 4

(Question facultative) Écrire une fonction `sauve_echiquier(n, N, nom_de_fichier)` produisant dans le fichier `nom_de_fichier` l'image d'un échiquier, où chaque case de l'échiquier a pour côté p pixels et N est le niveau d'intensité du blanc. Pour mémoire un échiquier a 64 cases, et dans sa représentation traditionnelle, la case en bas à droite est blanche.

Lecture et modification d'images

On pourra, pour tester les fonctions écrites ici, utiliser d'une part les images précédemment produites, d'autre part utiliser l'image disponible sur le site de la classe.

Question 5

Écrire une fonction `lit_image(nom_de_fichier)`, où la chaîne `nom_de_fichier` représente le nom d'un fichier PGM et renvoyant un couple `(img, N)` où `N` est le niveau d'intensité du blanc de l'image et `img` la matrice des pixels.

On suppose que le fichier respecte les contraintes données dans l'énoncé et on ne fera aucun effort particulier pour gérer les situations où il ne les respecterait pas.
Par exemple, votre fonction a le droit d'accepter un fichier dont les lignes font plus de 70 caractères. On suppose de plus que le fichier ne contient aucun commentaire.

R On pourra utiliser la fonction `lit_valeurs`, que vous trouverez sur le site de classe.

Question 6

Écrire une fonction `negatif(fichier_entree, fichier_sortie)` prenant en argument deux noms de fichiers `fichier_entree` et `fichier_sortie`. La fonction lit d'abord le fichier `fichier_entree` et crée le fichier `fichier_sortie` obtenu en remplaçant chaque pixel de niveau de gris k par un pixel de niveau $N-k$, où N est l'intensité du blanc du fichier d'entrée.

Question 7

(Question facultative) Écrire une fonction `rotation90(fichier_entr fichier_sortie)` lisant le fichier `fichier_entree` et créant le fichier `fichier_sortie` obtenu en effectuant une rotation de 90 degrés (dans le sens trigonométrique) de l'image originale.

Modéliser

2 Chaînes de caractères

Exercice 124 –

Écrire une fonction prenant en paramètre deux chaînes de caractères `texte` et `mot`, et recherchant la première occurrence de `mot` dans `texte` (la fonction retournera l'indice de la première lettre de l'occurrence de `mot` dans `texte`.

Exercice 125 –

Une chaîne de caractères $s_0 s_1 \dots s_{n-1}$ est un *palindrome* si elle est « symétrique » : $\forall k \in \{0,\dots,n-1\}, s_k = s_{n-1-k}$.

Question 1

Écrire une fonction `est_pal(s)` qui, à une chaîne de caractères `s`, renvoie le booléen `True` si `s` est un palindrome et `False` sinon.

Question 2

Écrire une fonction `max_pal(s)` qui, à une chaîne de caractères `s`, renvoie la chaîne `p` où `p` est la plus grande sous chaîne palindromique centrée (c'est la plus grande sous-chaîne palindromique de la forme $s_k s_{k+1} \dots s_{n-1-k}$).

Exercice 126 –

Question 1 *Écrire une fonction* `c_recherche(m,s)` *de recherche indiquant si le mot* `m` *est dans la chaîne* `s` *insensible à la casse (majuscules ou minuscules).*

Question 2 *Écrire une fonction* `i_recherche(m,s)` *de recherche du mot* `m` *dans une chaîne* `s`*, renvoyant la position de la première occurrence du mot dans la chaîne et une erreur si le mot n'est pas dans le texte.*

Exercice 127 –

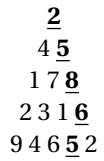
Question 1 *Écrire une fonction* `compte(m,s)` *comptant le nombre d'occurrences du mot* `m` *dans une chaîne* `s`*, chevauchements compris.*

Question 2 *Écrire une fonction* `dist_compte(m,s)` *comptant le nombre d'occurrences du mot* `m` *dans une chaîne* `s`*, sans chevauchements.*

3 Programmation dynamique

Exercice 128 –

Dans une pyramide de nombres, en partant du sommet, et en se dirigeant vers le bas à chaque étape, on cherche à maximiser le total de la somme des nombres traversés.

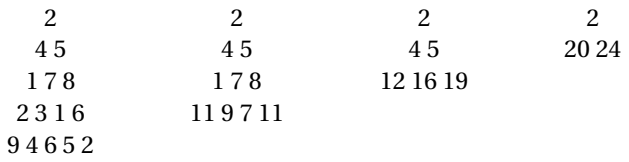


Sur cet exemple, la somme totale maximale est 26, obtenue en parcourant les nombres soulignés.

On peut voir la pyramide comme un graphe, parcourir les 16 chemins, et choisir celui qui a le plus grand total. Quand la pyramide a n niveaux, il y a 2^{n-1} chemins, ce qui rend vite cet algorithme inexploitable.

Un algorithme récursif est aussi possible, mais alors certains calculs sont effectués plusieurs fois, et là encore l'algorithme est trop long pour des pyramides de taille conséquente.

La méthode la plus rapide est celle utilisant la *programmation dynamique*. L'idée est résumée dans la séquence suivante :



À vous de comprendre cette idée et d'écrire un programme calculant ce total maximum pour des pyramides que l'on définit de la manière suivante : chaque pyramide est donnée dans un tableau, dont les éléments sont les lignes de la pyramide, représentées elles-mêmes dans un tableau. Par exemple, la pyramide de l'exemple sera représentée dans le tableau `[ [ 2 ], [ 4, 5 ], [ 1, 7, 8 ], [ 2, 3, 1, 6 ], [ 9, 4, 6, 5, 2 ] ]`.

Vous pourrez utiliser la fonction suivante pour construire ces pyramides :

```
def pyramide(alpha,n):
    p = []
    x = alpha
    for i in range(n):
        l = []
        for j in range(i+1) :
            y = x % 10
            l.append(y)
            x = (15091 * x) % 64007
        p.append(l)
    return p
```

Enfin, on rappelle que si $x \in \mathbb{N}$, on note $x\%10$ le reste de la division euclidienne par 10.

Question 1

Donner la somme maximale pour la pyramide ayant 20 lignes, définie par les $u_k\%10$ (lus de haut en bas, de gauche à droite, la première ligne est $[u_0\%10]$, la seconde $[u_1\%10, u_2\%10]$ *etc.*).

Question 2

Donner la somme maximale pour la pyramide ayant 100 lignes, définie par les $u_k\%10$ (lus de haut en bas, de gauche à droite, la première ligne est $[u_0\%10]$, la seconde $[u_1\%10, u_2\%10]$ *etc.*).

4 Complexité

Exercice 129 –

Dans ce problème, on considère des tableaux d'entiers relatifs `t=[t0,t1,...,tn-1]`, et on appelle *tranche* de `t` toute sous-tableau non vide `[ti,ti+1,...,tj-1]` d'entiers consécutifs de ce tableau (avec $0 \leq i < j \leq n$) qu'on notera désormais `t[i:j]`.
À toute tranche `t[i:j]` on associe la somme `s[i : j]` $= \sum_{k=i}^{j-1} t\_k$ des éléments qui la composent. Le but de ce problème est de déterminer un algorithme efficace pour déterminer la valeur minimale des sommes des tranches de `t`.

L'algorithme naïf

1. Définir une fonction `somme` prenant en paramètre un tableau `t` et deux entiers i et j, et retournant la somme `s[i : j]`.
2. En déduire une fonction `tranche_min1` prenant en paramètre un tableau `t` et retournant la somme minimale d'une tranche de `t`.
3. Montrer que la complexité de cet algorithme est en $\Theta(n^3)$, c'est-à-dire qu'il existe deux constantes $a, b \in \mathbb{R}_+^*$ telles que si `t` a n éléments, alors le nombre d'opérations effectuées dans le calcul de `tranche_min1(t)` est compris entre an^3 et bn^3.

Un algorithme de coût quadratique

1. Définir, sans utiliser la fonction `somme`, une fonction `mintranche` prenant en paramètres un tableau `t` et un entier `i`, et calculant la valeur minimale de la somme d'une tranche de `t` dont le premier élément est t_i , en parcourant une seule fois la liste a à partir de l'indice `i`.
2. En déduire une fonction `tranche_min2` permettant de déterminer la somme minimale des tranches de `t`, en temps quadratique, c'est-à-dire que la complexité de cet algorithme est en $\Theta(n^2)$. On justifiera que la complexité est précisément en $\Theta(n^2)$.

Un algorithme de coût linéaire

Étant donnée un tableau `t`, on note m_i la somme minimale d'une tranche quelconque du tableau `t[0 : i]`, et c_i la somme minimale d'une tranche de `t[0 : i]` se terminant par t_{i-1}.

Montrer que $c_{i+1} = \min(c_i + t_i, t_i)$ et $m_{i+1} = \min(m_i, c_{i+1})$, et en déduire une fonction `tranche_min3` de coût linéaire (c'est-à-dire dont la complexité est en $\Theta(n)$), calculant la somme minimale d'une tranche de `t`.

Exercice 130 –

Un enjeu scientifique et technologique actuel est de savoir traiter des problèmes mettant en jeu un nombre très important de données. Un point crucial est souvent de pouvoir manipuler des matrices de très grandes dimensions, ce qui est *a priori* très coûteux, en temps de calcul et en mémoire. On peut cependant souvent considérer que les matrices manipulées ne contiennent que « peu » d'éléments non nuls : c'est ce que l'on appelle les matrices *creuses*.

Nous nous intéressons ici à l'implémentation de deux algorithmes d'addition de matrices : l'un pour une représentation classique des matrices, l'autre pour une représentation des matrices creuses.

Soit $n \in \mathbb{N}^*$, on note $\mathcal{M}_n(\mathbb{R})$ l'ensemble des matrices carrées d'ordre n, à coefficients dans $\mathbb{R}$.

On représentera classiquement une matrice $M \in \mathcal{M}_n(\mathbb{R})$ par un tableau à double entrées. En `Python`, cela sera un tableau (type `list`) de longueur n, chaque élément de ce tableau représentant une ligne de M. Chaque élément de ce tableau est donc un tableau de longueur n, dont tous les éléments sont des nombres (types `int` ou `float`).

Cette même matrice M sera représentée de manière creuse en ne décrivant que ses cases non vides par un tableau de triplets `(i,j,x)`, où `x` est l'élément de M situé sur la i^e ligne et la j^e colonne. On pourra supposer que les éléments non nuls de M sont ainsi décrits ligne par ligne.

▪ **Exemple** La matrice $\begin{pmatrix} 1 & 0 & 5 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ sera représentée classiquement par le tableau

```
[ [1,0,5] , [0,-2,0] , [0,0,0] ]
```

et de manière creuse par le tableau

```
[ (0,0,1) , (0,2,5) , (1,1,-2) ].
```

▪

Question 1
Écrire une fonction `add(M,N)` prenant en argument deux représentations classiques `M` et `N` de deux matrices M et N (carrées, de même ordre) et renvoyant la représentation classique de $M+N$. On prendra soin d'écrire une fonction « optimale » en terme de complexité, spatiale et temporelle.

Question 2
Écrire une fonction `add_creuse(M,N)` prenant en argument deux représentations creuses `M` et `N` de deux matrices M et N (carrées, de même ordre) et renvoyant la représentation creuse de $M+N$. On prendra soin d'écrire une fonction « optimale » en terme de complexité, spatiale et temporelle.

Question 3
On suppose que M et N sont carrées, d'ordre n, représentées classiquement par `M` et `N`. Évaluer asymptotiquement la complexité temporelle de la fonction `add(M,N)`.

Question 4
On suppose que M et N sont carrées et contient chacune au plus p éléments non nuls, représentées de manière creuse par `M` et `N`. Évaluer asymptotiquement la complexité temporelle de la fonction `add_creuse(M,N)`.

Pour simplifier, on ne justifiera pas que les éléments obtenus sont disposés dans le bon ordre.

Question 5
Discuter du choix de la représentation pertinente à utiliser pour additionner deux matrices.

Exercice 131 –

On considère la suite u à valeurs dans $[\![0;64\,007[\![$ définie par

$$u_0 = 42, \quad \forall n \in \mathbb{N},\ u_{n+1} = 15\,091\,u_n \ [64\,007],$$

ainsi que, pour tout $n \in \mathbb{N}$, $S_n = \sum_{k=0}^{n} u_k$.

On propose l'algorithme suivant pour calculer les valeurs de S.

```
def u(n):
    """u_n, n : entier naturel"""
    v = 42
    # Inv : v = u_0
    for k in range(n):
        # Inv : v = u_k
        v = 15091 * v % 64007
        # Inv : v = 15091*u_k % 64007 = u_k+1
    # Inv : au dernier tour, k = n-1, donc v =
        u_n
    return v

def S(n):
    """u_n, n : entier naturel"""
    s = u(0)
    # Inv : s = S_0
    for k in range(n):
        # Inv : s = S_k
        s = s + u(k+1)
        # Inv : v = S_k+u_k+1 = S_k+1
    # Inv : au dernier tour, k = n-1, donc s =
        S_n
    return s
```

1. Étudier les complexités des fonctions `u` et `S`, en fonction de n.
2. Écrire une fonction donnant la valeur de S_n en temps $O(n)$.

Exercice 132 –

On considère la suite u définie par

$$u_0 = 2, \quad \forall n \in \mathbb{N},\ u_{n+1} = u_n^2.$$

On se considère la fonction suivante, permettant de calculer les valeurs de u.

```
def u(n):
    """u_n, n : entier naturel"""
    v = 2
    # Inv : v = u_0
    for k in range(n):
        # Inv : v = u_k
        v = v*v
        # Inv : v = u_k**2 = u_k+1
    # Inv : au dernier tour,
    # k = n-1, donc v = u_n
    return v
```

Pour étudier le temps d'exécution d'une fonction, on pourra utiliser le morceau de code suivant.

```
import timeit

REPEAT=3

def duree(f, x):
  """Calcule le temps mis par Python pour
  calculer f(x). Cette fonction effectue
  en fait le calcul de f(x) REPEAT fois et
  garde la valeur la plus petite (l'idée est
  d'éliminer les éventuelles perturbations
  provoquées par d'autres processus
  tournant sur la machine)"""
  t = timeit.Timer(stmt=lambda : f(x))
  time = min(t.repeat(REPEAT, number=1))
  return time
```

1. Étudier en fonction de n la complexité asymptotique de la fonction `u`, dans le modèle standard.
2. Tracer les temps de calculs de u_k pour $k \in [\![0;30[\![$ par la fonction `u`. Discuter le résultat.
   Indication : on pourra utiliser une échelle semi-logarithmique.
3. Proposer un modèle de complexité plus réaliste et étudier dans ce modèle n la complexité asymptotique de la fonction `u`. On pourra déterminer explicitement u_n.

Exercice 133 –

On considère la fonction Python suivante.

```
def mystere(t):
    """t : tableau d'entiers"""
    m = 0
    for i in range(len(t)):
        for j in range(i+1,len(t)):
            if t[j] - t [i] > m :
                m = t[j]-t[i]
    return m
```

Question 1

Que contient le résultat renvoyé par `mystere(t)` ? Justifier.

Question 2

Quelle est la complexité de la fonction `mystere(t)`, en fonction de la longueur de `t`, que l'on notera n ?

Exercice 134 –

On considère la fonction Python suivante.

```
def mystere(u,v):
    """u,v : tableaux d'entiers"""
    m = 0
    p,q = len(u),len(v)
    for i in range(p) :
        for a in range(q) :
            k = 0
            while i+k < p and a+k < q and u[i+k]
                == v[a+k] :
                k = k+1
            if k > m :
                m = k
    return m
```

Question 1

Que contient le résultat renvoyé par `mystere(u,v)` ? Justifier.

Question 2

Quelle est la complexité de la fonction `mystere(u,v)`, en fonction du maximum des longueurs de `u` et `v`, que l'on notera n ?

Exercice 135 –

On considère le code suivant, qui crée une liste aléatoire `L` et qui la trie ensuite par ordre croissant par la méthode du «tri par insertion».

Modéliser

```
def liste_triee(n):
    """Renvoie une liste d'éléments de range
        (100), de longueur n,
       triée par ordre croissant """
    L = []
    for i in range(n):
        L.append(randrange(100))
    for i in range(1,n):
        # Inv : L[:i] est triée par ordre
            croissant
        # Idée : on fait redescendre L[i] pour l
            'insérer au bon endroit
        j = i
        while j >= 1 and L[j] < L[j-1]:
            # On échange L[j-1] et L[j]
            L[j], L[j-1] = L[j-1], L[j]
            j = j-1
    return L
```

Un appel de `randrange(100)` renvoie un nombre tiré aléatoirement et uniformément dans $[\![0,100[\![$ et a une complexité en $O(1)$.

Question 1

Peut-on donner explicitement et exactement une complexité pour la boucle `while` des lignes 13 à 16? Que peut-on quand même proposer comme type de complexité pour cette boucle?

Donner dans ce cas la complexité de cette boucle en fonction de `i`.

Question 2

Donner dans ce cas la complexité d'un appel de `liste_triee(n)` en fonction de `n`.

Exercice 136 –

Beaucoup de problèmes technologiques actuels mettent en jeu des données de grandes dimensions et font souvent intervenir de grandes matrices.

Beaucoup de ces matrices sont *creuses*, c'est-à-dire qu'elles ne contiennent que peu de valeurs non nulles. Nous allons développer une méthode de codage de telles matrices et étudier leur intérêt.

Dans tout ce problème, on s'interdira d'utiliser les opérations entre vecteurs et matrices `numpy` (méthode `.dot()` par exemple).

Dans tout ce problème, on utilisera le type `array` de `numpy` pour représenter des vecteurs. On pourra supposer que la ligne suivante a été écrite :

```
from numpy import array, zeros
```

Dans tout ce problème, on veillera à l'optimalité des fonctions écrites en termes de complexité temporelle asymptotique. Les réponses clairement sous-optimales seront pénalisées.

Soit une matrice réelle $A \in \mathcal{M}_{n,p}(\mathbb{R})$ de dimension $n \times p$ et possédant s coefficients non nuls. Ce coefficient s est appelé *niveau de remplissage* de la matrice A. Comme toujours en `Python`, on numérotera les lignes et les colonnes en partant de 0.

Une telle matrice se code usuellement comme un tableau de nombres de dimension $n \times p$.

Le codage CSR (Compress Sparse Raw) code la matrice A par trois listes V, L et C définies comme suit.

- La liste V contient toutes les valeurs des coefficients non nuls de A, en les listant ligne par ligne (de la première à la dernière ligne puis de la première à la dernière colonne). Ainsi, V est de longueur s.
- La liste L est de longueur $n+1$, L_0 vaut toujours 0 et pour chaque $1 \leq i \leq n$, L_i vaut le nombre de coefficients non nuls dans les i premières lignes de A (*i.e.* de la ligne d'indice 0 à celle d'indice $i-1$, inclu).
- La liste C contient les numéros de colonne de chaque coefficients non nuls de A, en les listant ligne par ligne (de la première à la dernière ligne puis de la première à la dernière colonne). Ainsi, C est de longueur s.

Par exemple, avec

$$A = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 2 & 0 & 4 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

on a $n=3$, $p=4$, $s=4$ et sa représentation CSR est donnée par les trois listes

$$V = [-1,2,4,-1],$$
$$L = [0,1,3,4],$$
$$C = [1,0,2,3].$$

On remarquera qu'ajouter une colonne nulle à droite de A ne change pas sa représentation CSR.

Question 1

Déterminer la représentation CSR de la matrice

$$A = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Question 2

Donner une matrice dont la représentation CSR est

$$V = [2,-3,1,1],$$
$$L = [0,0,2,4],$$
$$C = [0,3,2,3].$$

Question 3

Soit (V,L,C) la représentation CSR de A, soit $0 \leq i \leq n-1$. Combien y a-t-il de coefficients non nuls sur la ligne n° i de A? Donner deux expressions `Python` (en fonction de V, L, C et i) permettant d'obtenir respectivement la liste des valeurs de ces coefficients et la liste des indices colonnes de ces coefficients.

On rappelle la formule du produit matriciel : pour une matrice $A \in \mathcal{M}_{n,p}(\mathbb{R})$ de coefficients $a_{i,j}$ et un vecteur $X \in \mathcal{M}_{p,1}(\mathbb{R})$ de coefficients x_j, si $0 \leq i \leq n-1$, alors la i^{e} coordonnée de AX est

$$\sum_{j=0}^{p-1} a_{i,j} x_j.$$

Question 4

Écrire une fonction `coeff_prod(V,L,C,X,i)` renvoyant la i^{e} coordonnée du produit AX, où le triplet

(V, L, C) est la représentation CSR de A. On supposera que les dimension de A et X sont compatibles pour effectuer le produit AX.

Donner la complexité asymptotique de cette fonction, en fonction de n, p et ℓ_i, où ℓ_i est le nombre d'éléments non nuls sur la i^e ligne de A.

Question 5

Écrire une fonction `prod(V,L,C,X)` renvoyant le produit AX, où le triplet (V, L, C) est la représentation CSR de A. On supposera que les dimension de A et X sont compatibles pour effectuer le produit AX.

Donner la complexité asymptotique de cette fonction, en fonction de n, p et s.

Question 6

Écrire une fonction `prod_naif(A,X)` renvoyant le produit AX, où A est codée usuellement comme un tableau à double dimension.

Donner la complexité asymptotique de cette fonction, en fonction de n, et p.

Question 7

En les comparant, discuter des deux complexités précédentes, notamment en fonction du niveau de remplissage s.

On prend maintenant pour exemple celui de la dérivation discrète, typique en traitement du signal. Pour un vecteur de longueur n

$$X = \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

on définit la dérivée discrète de X comme le vecteur Y de longueur n définit par :

$$y_0 = x_1 - x_0,\ y_{n-1} = x_{n-1} - x_{n-2} \text{ et } \forall i \in [\![1, n-1[\![,\ y_i = \frac{1}{2}(x_{i+1} - x_{i-1}).$$

Question 8

Déterminer une matrice A vérifiant $Y = AX$.

Question 9

Écrire une fonction `CSR_A(n)` renvoyant les trois vecteurs `V`, `L` et `C` codant A au format CSR.

Indication : on pourra écrire une fonction pour la création de chaque vecteur

Exercice 137 –

Question 1 *Étudier la complexité théorique de la fonction* `maxi`

```
def maxi(t):
    """Renvoie le plus grand élément de t.
       Précondition : t est un tableau non vide
           """
    m = t[0]
    for x in t:
        # Invariant : m est le plus grand élé
            ment trouvé jusqu'ici
        if x > m:
            m = x # On a trouvé plus grand, on
                met àjour m
    return m
```

Exercice 138 –

Question 1 *Étudier les complexité théoriques (dans le pire des cas) des fonctions* `appartient` *et* `appartient_dicho`. *Les comparer.*

```
def appartient(e, t):
    """Renvoie un booléen disant si e appartient
         àt
       Précondition : t est un tableau"""
    for x in t:
        # Invariant : e n'est pas positionné
            dans t avant x
        if e == x:
            return True # On a trouvé e, on s'
                arrÃªte
    return False

def appartient_dicho(e,t):
    """Renvoie un booléen indiquant si e est
        dans t
       Préconditions : t est un tableau de
           nombres trié par ordre croissant
                   e est un nombre"""
    g = 0 # Limite gauche de la tranche où l'on
        recherche e
    d = len(t)-1 # Limite droite de la tranche o
        ù l'on recherche e
    while g <= d: # La tranche où l'on cherche e
         n'est pas vide
        m = (g+d)//2 # Milieu de la tranche où l
            'on recherche e
        pivot = t[m]
        if e == pivot: # On a trouvé e
            return True
        elif e < pivot:
            d = m-1 # On recherche e dans la
                partie gauche de la tranche
        else:
            g = m+1 # On recherche e dans la
                partie droite de la tranche
    return False
```

Exercice 139 –

Question 1 *Étudier la complexité théorique dans le pire des cas de la fonction* `recherche`. *On pourra être amené à la reformuler légèrement.*

```
def recherche(m,s):
    """Recherche le mot m dans la chaÃ®ne s
       Préconditions : m et s sont des chaÃ®nes
           de caractères"""
    long_s = len(s) # Longueur de s
    long_m = len(m) # Longueur de m
    for i in range(long_s-long_m+1):
        # Invariant : m n'a pas été trouvé dans
            s[0:i+long_m-1]
        if s[i:i+long_m] == m: # On a trouvé m
            return True
    return False
```

Exercice 140 –

Question 1 *Étudier la complexité théorique de la fonction* `conv_b2`

```
def conv_b2(p):
```

```
    """Convertit l'entier p en base 2 (renvoie une
        chaÃ®ne)"""
    x = p
    s = ""
    while x > 1 :
        s = str(x%2) + s
        x = x // 2
    return str(x)+s
```

Question 2 *Étudier les complexités théoriques des fonctions* `calc_b2_naif` *et* `calc_b2_horner`. *Les comparer.*

```
def calc_b2_naif(s):
    """Renvoie l'entier p représente en binaire
        par s"""
    p = 0
    x = 1 ## 2**0
    for i in range(len(s)):
        p = p+int(s[len(s)-i-1])*x
        x = 2*x
    return p

def calc_b2_horner(s):
    """Renvoie l'entier p représente en binaire
        par s"""
    p = int(s[0])
    for i in range(1,len(s)):
        p = int(s[i])+2*p
    return p
```

5 Fichiers

Exercice 141 – (FIC-000)

Sur un réseau social, vous avez un certain nombre d'amis. Vous pouvez aller sur la page de vos amis afin de savoir combien ils ont d'amis chacuns. Le réseau va également vous indiquer combien d'amis vous avez en commun avec chacun de vos amis. Cependant, vous aimeriez en savoir un peu plus. Plus précisément, vous vous demandez combien il y a de gens qui font partie des amis de vos amis et qui ne font pas déjà partie de vos amis.

Compter à la main prendrait bien trop de temps, donc vous décidez d'écrire un algorithme pour cela. Votre algorithme utilisera comme données le fichier `amis.csv`, dont les lignes comportent deux entiers chacune, séparés par une virgule, qui sont deux identifiants de personnes amies l'une de l'autre.

Notez que l'amitié est toujours réciproque : si I_1 est ami de I_2 alors I_2 est ami de I_1. On donnera soit (I_1, I_2) soit (I_2, I_1) dans le fichier, mais pas les deux. De plus, une personne n'est jamais amie d'elle-même, *i.e.* I_1 et I_2 sont toujours différents.

Prenons comme exemple le fichier suivant.

```
0,1
0,2
0,3
1,2
1,3
2,3
1,4
2,5
2,4
3,6
7,8
```

Alors, les amis de la personne d'identifiant 0 sont les personnes d'identifiants 1, 2 et 3. Les amis des personnes d'identifiants 1, 2 et 3 sont les personnes d'identifiants 0, 1, 2, 3, 4, 5 et 6. Les amis d'amis de la personne d'identifiant 0 qui ne sont pas des amis directs de la personne d'identifiant 0 (ni cette même personne) sont donc les personnes d'identifiant 4, 5 et 6. Il y en a donc 3.

Question 1

Combien d'amis d'amis de la personne d'identifiant α ne sont pas des amis de la personne d'identifiant α, ni α ?

Exercice 142 – (FIC-001)

Question 1

Écrire une fonction `resume(nom_de_fichier)` qui prend en argument une chaîne de caractères `nom_de_fichier` et qui renvoie le triplet `(L,m,c)` où, pour le fichier dont le chemin est `nom_de_fichier`; :

- textttL est le nombre de lignes du fichier ;
- textttm est le nombre de mots (sous chaîne maximale non vide de caractères consécutifs, sans blanc) du fichier ;
- texttc est le nombre de caractères du fichier (blancs compris).

On rappelle qu'un blanc est un retour chariot (`\n`), une tabulation (`\t`) ou une espace.

Exercice 143 – (FIC-002)

On suppose que l'on dispose d'un fichier `nombres.txt` contenant des entiers naturels séparés pas des blancs (on rappelle qu'un blanc est soit un espace, soit une tabulation `\t` soit un retour à la ligne `\n`).

Pour faciliter votre travail, un exemple de tel fichier est disponible sur le site de classe. Nous vous encourageons fortement à tester vos fonctions sur des exemples dont vous aurez calculé le résultat à la main.

Question 1

Écrire une fonction `somme()`, sans argument, renvoyant la somme de tous les entiers contenus dans ce fichier.

Question 2

Pour chaque ligne du fichier, on effectue le produit des entiers de cette ligne. Écrire une fonction `moyenne()`, sans argument, renvoyant la moyenne de tous ces produits.

Question 3

Si i est un entier inférieur au nombre de lignes du fichier, on note s_i la somme des entiers de la ligne i. Écrire une fonction `inversion()`, sans argument, retournant le nombre de couples (i, j) tels que $i < j$, et $s_j < s_i$.

Les invariants de boucle seront impérativement précisés en commentaires, dans le script renvoyé.

Question 4

Dans cette dernière question, on suppose que le fichier contient n lignes, et que chaque ligne ne contient qu'un entier. Quelle est en fonction de n la complexité de l'algorithme `inversion()` ?

On supposera que les opérations de lecture dans le fichier se font en temps constant. Ainsi, l'opération consistant à lire tous les entiers du fichier et à les stocker un par un dans un tableau sera supposée avoir une complexité en $O(n)$.

Pour plus de clarté, vous recopierez le code de la fonction `inversion()` avant d'étudier sa complexité.

Exercice 144 – (FIC-003)

Le fichier `gilberte.txt` contient un extrait du premier volume d'*À l'ombre des jeunes filles en fleurs* de Marcel Proust, où il évoque la fin de sa relation avec Gilberte Swann, son premier amour.

Dans toute la suite, les questions porteront sur les lignes allant du numéro α inclus au numéro $\alpha+300$ exclu. Comme toujours en `Python`, la première ligne du texte porte le numéro 0.

On pourra remarquer qu'aucun mot n'est coupé lors d'un retour à la ligne, ce qui simplifiera les choses.

Question 1

Combien de fois apparaît le mot « Gilberte » ?

On appellera ici *paragraphe* toute partie du texte telle que :

- le premier caractère est une tabulation ;
- le caractère précédant cette tabulation est un retour à la ligne, sauf s'il s'agit du début du texte ;
- le dernier caractère est un retour à la ligne ;

- le caractère suivant ce retour à la ligne est une tabulation, sauf s'il s'agit de la fin du texte.

Question 2
Combien y a-t-il de paragraphes ?
Proust est célèbre pour la longueur de ces phrases. Vous allez donc chercher la phrase la plus longue de la portion du texte que vous devez étudier.
On rappelle que l'on appelle *blanc* un caractère qui est un espace, un retour chariot ou une tabulation.
On appellera *point* les caractères « . », «! » et «? ».
On appellera ici *phrase* toute portion du texte telle que :

- le premier caractère est un blanc ;
- le caractère précédant ce blanc est un retour à la ligne ou un point, sauf s'il s'agit du début du texte ;
- le dernier caractère est un point ;
- le caractère suivant ce point est un blanc, sauf s'il s'agit de la fin du texte.

Attention, une phrase peut contenir un point autre que son dernier caractère, par exemple dans les phrases :

Je m'écriai « Gilberte ! » en la voyant.

Ou

J'étais perplexe ...

Un autre exemple pour être sûr que tout est compris : dans le texte :

Bonjour. Ça va ?
Comment te portes-tu ?
Très bien,
je te remercie.

il y a quatre phrases, la phrase la plus longue est la dernière, et elle comporte 28 caractères (ne pas oublier les deux blancs au début (retour chariot et tabulation), le retour chariot au milieu et le point à la fin !).

Et une dernière remarque : votre portion de texte risque de commencer au milieu d'une phrase et de finir au milieu d'une autre : pour simplifier les choses, la portion de votre texte commençant au premier caractère de votre texte et finissant là où commence la phrase suivante ne sera pas prise en compte, même si par hasard c'était une phrase complète. Et si votre texte ne finit pas par un point suivi d'un blanc, vous ne tiendrez pas compte de la phrase incomplète qui clôt votre portion de texte.

Question 3
Quel est le nombre de caractères de la phrase la plus longue ?

Question 4
Quel est le nombre de caractères de la phrase la plus courte ?
Le texte d'*À la recherche du temps perdu* étant tombé dans le domaine public depuis 1987, vous décidez de publier vous-mêmes la partie du texte qui vous est échue aujourd'hui, dans une édition au format particulier : ayant hérité de votre oncle épicier d'une grande quantité de rouleaux pour caisse enregistreuse, vous décidez de faire imprimer le texte sur ce papier. Dans un souci de lisibilité, vous décidez également d'imprimer ce texte dans une police de taille standard : chaque ligne comportera au plus neuf caractères, sans compter les retours chariot.
La règle est la suivante : les retours à la ligne du texte de base sont tous supprimés, sauf ceux marquant un changement de paragraphe. Les retours à la ligne de la nouvelle édition seront donc ceux des changements de paragraphe et ceux imposés par la taille maximale de neuf caractères de chaque ligne. S'il le faut, les mots seront coupés par un retour à la ligne sans scrupule. Par contre, les espaces en début de ligne seront supprimés.
Par exemple, le texte :

Marcel Proust est un enfant à la santé fragile. Toute sa vie il a des difficultés respiratoires.
Très jeune, il fréquente des salons aristocratiques.

sera réédité de la manière suivante :

Marcel P
roust est
un enfant
à la sant
é fragile
. Toute s
a vie il
a des dif
ficultés
respirato
ires.
Très jeu
ne, il fr
équente d
es salons
aristocra
tiques.

Question 5
Donner la 71-ème ligne (donc la ligne numéro 70) de votre texte réédité : on signalera les tabulations en début de ligne et les espaces par le symbole.

Exercice 145 – (FIC-004)

Question 1 *Écrire une fonction* `moyennes(fichier_notes,fichi` *lisant les notes écrites dans* `fichier_notes` *et écrivant dans le* `fichier_moyennes` *le prénom de chaque élève, suivi par la moyenne de ses notes.*

Par convention, la première ligne de chaque fichier comporte les titres des colonnes et la première colonne contient le prénom de chaque étudiant. Un exemple (fichiers `notes.csv` et `moyennes.csv`) sera mis en ligne sur le site de classe.

On respectera le format `.csv`, le séparateur par défaut sera donc les virgules (`','`).

Exercice 146 – (FIC-005)

Analyse préliminaire : filtrage numérique

Filtre numérique passe bas du premier ordre

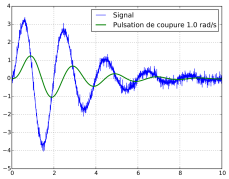
Soit un filtre linéaire du premier ordre. Son équation différentielle est de la forme :

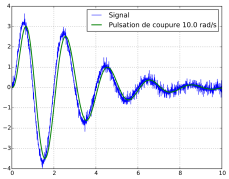
$$s(t)+\tau\frac{\mathrm{d}s(t)}{\mathrm{d}t}=Ke(t)$$

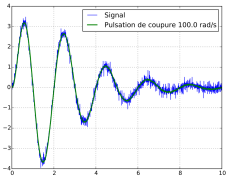
En utilisant un schéma d'Euler implicite, on a l'approximation suivante : $\frac{\mathrm{d}s(t)}{\mathrm{d}t}=\frac{s_k-s_{k-1}}{h}$. En conséquences,

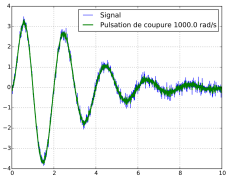
$$s_k+\tau\frac{s_k-s_{k-1}}{h}=Ke_k \Leftrightarrow s_k=\frac{hKe_k+\tau s_{k-1}}{h+\tau}$$

La fréquence d'échantillonnage est ici de 1 KHz. Le pas de dérivation est de 0,001 s. On réalise alors différents filtrages en faisant varier la pulsation de coupure du filtre.









On observe que lorsque la pulsation de coupure diminue, le signal est de plus en plus lissé, limitant ainsi le bruit. En revanche, le signal est atténue et déphasé.

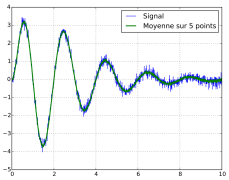
On donne l'algorithme permettant de réaliser ce filtre à partir d'un tableau "$signal$" contenant des valeurs pour chaque instant t correspondant au tableau "t" avec une fréquence de coupure "$freq$".

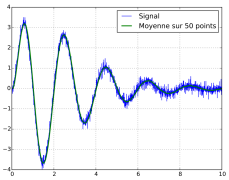
```
def filtrage_passe_bas(freq,t,signal):
    tau = 1/freq # Coupure du filtre
    K=1
    res=[signal[0]]
    for i in range(1,len(signal)):
        h=t[i]-t[i-1]
        res.append((h*K*signal[i]+tau*res[-1])/(
            h+tau))
    return res
```

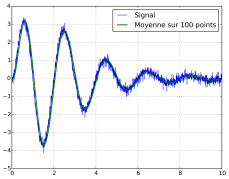
Filtrage numérique par moyenne glissante

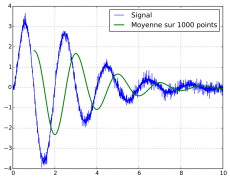
Une autre solution pour lisser une courbe est de réaliser une moyenne glissante. Si on réalise ce filtrage en temps réel, cela signifie qu'un point lissé à l'échantillon n est la moyenne des $n-1$ échantillons précédents et de l'élément en cours.

Il faut donc attendre l'acquisition de $n-1$ échantillons avant de disposer de la courbe.









On donne l'algorithme permettant de réaliser ce filtre à partir d'un tableau "$signal$" contenant des valeurs pour chaque instant t correspondant au tableau "t" avec une taille de fenêtre de filtrage "$freq$".

```
def filtrage_moyenne(signal,fenetre):
    filtrageg =signal[0:fenetre-1]
    for i in range(fenetre-1,len(signal)):
        s = sum(signal[i-fenetre+1:i+1])/fenetre
        filtrageg=np.concatenate([filtrageg,np.
            array([s])])
    return filtrageg
```

Références

- Cours de Xavier Pessoles support de cours de PSI* La Martinière Monplaisir.
- Patrick Beynet, *Supports de cours de TSI 2*, Lycée Rouvière, Toulon.
- David Crochet (créer à partir de KmPlot), CC-BY-SA-3.0, via Wikimedia Commons `https://upload.wikimedia.org/wikipedia/commons/6/6f/Fourier_d%27un_carr%C3%A9.svg`.

Analyse des performances d'un cycliste

Analyse d'un fichier gpx

Lecture du fichier

Question 1

Ecrire un programme permettant de savoir si un mot se trouve dans une chaine de caractère.

Un fichier gpx permet d'écrire l'ensemble des données issues d'un appareil du type gps ou montre connecté. Il recueille les 3 coordonnées spatiales (altitude par rapport au niveau de la mer, latitude et longitude) en fonction du temps. Ce fichier contient donc un ensemble de coordonnées spatiales en fonction du temps. Ouvrir le fichier gpx et analyser son contenu. On remarquera en particulier que le fichier est organisé d'un certaine manière. Ce fichier comporte différentes balises [5].

Le fichier ($< gpx >$) peut contenir :

- Des métadonnées ($< metadata >$), décrivant le contenu du fichier GPX par entre autre :
  - un nom ($< name >$)
  - une description ($< desc >$)
  - l'auteur du fichier ($< author >$) comprenant son nom, une adresse mail et un lien vers son site web.
  - ...
- Une liste de traces ou track ($< trk >$) chacune décrite par :
  - un nom ($< name >$) ;
  - le type d'itinéraire($< type >$)

5. source : wikipedia : `https://fr.wikipedia.org/wiki/GPX_(format_de_fichier)`

Modéliser

- des segments de trace (< $trkseg$ >), le passage d'un segment à un autre indique une extinction du récepteur GPS ou une perte de réception. Un segment de trace est constitué :
  * d'une liste ordonnée de points de trace (< $trkpt$ >) dans laquelle figure respectivement la latitude ('lat') ainsi que la longitude ('long') en °.
  * avec son altitude par rapport au niveau de la mer en mètres (< ele >)
  * un horodatage (< $time$ >)

Dans ce problème on pourra travailler sur un des deux fichiers gpx donné sur le site de la classe dans "info/mpsi2/Programmes montrés dans le cours"(rumilly.gpx, mont_dor.gpx).

Question 2

Ecrire un programme permettant de stocker sous la forme de 4 tableaux nommés tp, lat, $long$ et lat représentant respectivement pour chaque point de mesure le temps en s par rapport à l'instant initial, la latitude en °, la longitude en ° ainsi que l'altitude par rapport au niveau de la mer en m.

Superposition du tracé sur une carte

Pour des traces petites (comme un parcours cycliste) on peu assimiler les longitude et latitude à des coordonnées cartésiennes et il n'est donc pas nécessaire de réaliser des projections cartographiques.

Question 3

A l'aide des fonctions contenues dans le module "$matplotlib.pyplot$" tracer un graphe décrivant la liste des points des différentes coordonnées extraites précédemment en mettant en abscisses les valeurs de longitude et en ordonnées celles de latitude. On obtient alors la trace gps du parcours réalisé.

Pour superposer cette trace à une carte, on peut utiliser le module "$folium$" qu'il faut installer avec la commande :

```
pip install folium
```

Puis importer avec :

```
import folium
```

La fonction "Map" permet de créer un objet carte centrée avec en point possédant une latitude et une longitude. Dans les arguments de cette fonction, il faut préciser les coordonnées en latitude et en longitude (en °) ainsi le niveau d'agrandissement initial avec l'entier z.

```
macarte = folium.Map(location=[lat,long],
    zoom_start=z)
```

Etant donnée une trace GPS comme celle importée précédemment, on peut connaître les latitudes min,max et les longitudes min,max qui lui correspondent. On peut se servir de ces dernière pour définir la carte initiale.

Pour superposer à cette carte la trace gps extraite précédemment on peut utiliser la fonction "$PolyLine$" qui est utilisable de la façon suivante :

```
folium.PolyLine(points).add_to(macarte)
```

Où la variable "$points$" est une liste de tuple contenant pour chaque coordonnée du fichier gpx respectivement la latitude et la longitude.

La méthode "save" appliqué à l'objet "macarte" et prenant en argument une chaine de caractère correspondante au nom du fichier "html" permet de créer une page html représentant la carte définie précédemment.

Question 4

mettre en oeuvre les fonctions du module "$folium$" pour créer une page HTML superposant la trace GPS sur une carte avec un centrage et un niveau d'agrandissement satisfaisant. Vous pourrez visualiser la page créée en l'ouvrant un navigateur internet (Firefox, Google Chrome, etc...)

Traitement du fichier

On donne l'expression de la vitesse et de l'accélération en coordonnées sphérique d'un point matériel M dans son mouvement par rapport au référentiel terrestre R_0 :

$$\overrightarrow{V}(M/R_0) = (\dot{r})\,\overrightarrow{e}_r + \left(r\cdot\dot{\theta}\right)\overrightarrow{e}_\theta + \left(r\cdot\sin\theta\,\dot{\varphi}\right)\overrightarrow{e}_\varphi$$

$$\overrightarrow{a}(M/R_0) = \left(\ddot{r} - r\cdot\dot{\theta}^2 - r\cdot\sin^2\theta\,\dot{\varphi}^2\right)\overrightarrow{e}_r + \left(2*\dot{r}\dot{\theta} + r\cdot\ddot{\theta} - r\cdot\sin\theta$$
$$+\left(2r\cos\theta\,\dot{\theta}\,\dot{\varphi} + 2\dot{r}\sin\theta\,\dot{\varphi}\right.$$

On peut relier les paramètres r, θ et φ aux coordonnées gps :

$$\begin{cases} r = alt + R \\ \theta = \dfrac{\pi}{2} - lat \\ \varphi = long \end{cases}$$

avec R le rayon de la terre supposé constant et égal à $6371km$

Question 5

Écrire un programme prenant en argument un vecteur $y(t)$ dépendant du temps ainsi qu'un vecteur t de mêmes longueurs et renvoyant le vecteur dy correspondant à l'estimation de $\dfrac{dy(t)}{dt}$ par différence finie. On pourra la noter $diff(y,t)$.

Question 6

Appliquer cette méthode pour donner des estimation de $\dot{r}$, $\dot{\theta}$, $\dot{\varphi}$ que pouvons-nous dire quant à la qualité de l'estimation de la dérivée?

Question 7

Utiliser les fonctions de filtrage numérique décrites en première partie pour améliorer les résultats.

Question 8

Écrire une fonction $vitesse$ prenant en argument les tableaux r, $theta$ et phi et renvoyant 4 tableau Vr, $Vtheta$ et $Vphi$ et V correspondant respectivement aux 3 coordonnées du vecteur vitesse dans le système de coordonnées sphérique ainsi que la norme du vecteur vitesse.

Question 9
Mettre en évidence à l'aide d'un tracé l'éventuelle corrélation entre la vitesse du cycliste et le profil en altitude du circuit.

Modélisation des performances mécaniques

Modélisation de la puissance

En considérant l'ensemble (cycliste+velo) comme un solide à masse ponctuelle (cela revient à négliger les effets d'inertie de la rotation des roues) on peut estimer la puissance que le cycliste doit développer en mouvement (P_c) à l'aide du théorème de l'énergie cinétique :

$$P_c = P_{inertie} + P_{aero} + P_{Pes} + P_{roul}$$

- $P_{inertie}$ est la puissance due aux effets d'inertie et elle provient de la variation de l'énergie cinétique.
- P_{aero} est la puissance qui résulte des effets aéronautiques.
- P_{pes} est la puissance qui provient de l'action mécanique de pesanteur.
- P_{roul} est la puissance due à la résistance au roulement.

On se placera dans le référentiel terrestre supposé galiléen (R_0). Le cycliste ayant réalisé l'activité possède une masse $M = 70kg$. On prendra l'accélération de la pesanteur $g = 9,81m \cdot s^{-2}$.

Puissance due à la résistance au roulement

Question 10
Déterminer une fonction P_{roul} qui prend en argument la masse d'un cycliste, un vecteur V comprenant la norme de la vitesse de déplacement du cycliste ($V = \| \overrightarrow{V}(M/R_0) \|$) associé à une trace GPX, la pression dans les pneus et qui renvoie la puissance instantanée de roulement.

Puissance due aux effets d'inertie

La puissance due aux effets d'inertie peut s'estimer de la façon suivante :

$$P_{inertie} = M \cdot V \cdot \frac{dV}{dt}$$

Question 11
Déterminer une fonction $Pinertie$ prenant en arguments la masse d'un cycliste, un vecteur V comprenant la norme de la vitesse de déplacement du cycliste ($\| \overrightarrow{V}(M/R_0) \|$) associé à une trace GPX, la pression dans les pneus et qui renvoie la puissance instantanée de roulement.

Puissance aéronautique

Question 12
Déterminer une fonction $Paero$ prenant en arguments, la vitesse du vent (Vv),sa direction (α_v), le produit $C_d \times A_p$, un vecteur V comprenant les composantes de la vitesse de déplacement du cycliste ($V(M/R_0)$) en coordonnées sphériques associées à une trace GPX et renvoyant l'estimation de la puissance aérodynamique instantanée.

Puissance due aux effets de pesanteur

La puissance de pesanteur va dépendre du dénivelé qui correspond à la variation de la pente en fonction de la distance parcouru.

Question 13
Écrire une fonction permettant de calculer l'intégrale par rapport au temps d'une fonction $f(t)$ avec la méthode des trapèze. Elle prendra en arguments deux tableaux de même dimension f et t et renverra l'estimation de $I = \int_{t'=0}^{t} f(t')dt'$ sous la forme d'un tableau ayant la même dimension que t et f.

Question 14
Utiliser cette fonction pour donner une estimation de la distance parcouru en fonction du temps que l'on notera x.

La pente instantanée peut s'estimer par $\frac{dr(t)}{dx}$.

Question 15
Écrire une fonction prenant en argument les tableaux r et x et renvoyant l'estimation de la pente sous la forme d'un tableau de même dimension. Pour vérifier la cohérence de cette estimation, tracer sur une même figure deux graphes représentant en fonction du temps $r(t)$ et $\frac{dr(t)}{dx}$.

La puissance de pesanteur peut s'estimer de la façon suivante :

$$P_{pes} = M \cdot g \cdot \sin\left[\arctan\left(\frac{dr(t)}{dx}\right)\right] \cdot \| \overrightarrow{V}(M/R_0) \|$$

Question 16
Déterminer une fonction $Ppes$ prenant en arguments, la masse du cycliste, la coordonnées $r(t)$, un vecteur V comprenant la norme de la vitesse de déplacement du cycliste ($\| \overrightarrow{V}(M/R_0) \|$) associé à une trace GPX et renvoyant l'estimation de la puissance de pesanteur instantanée.

Puissance totale et profile de puissance record

Puissance totale

Question 17
A l'aide des questions précédentes tracer en fonction du temps les différentes puissances instantanées. Estimer la puissance totale instantanée que doit fournir le cycliste (Pc) et la superposer sur ce graphe.

Question 18
Comment interpréter les valeurs négatives. On veillera à ne pas en tenir compte pour la suite.

L'énergie dépensée sur le circuit par le cycliste peut s'estimer de la façon suivante :

$$E = \int_{t=0}^{t_f} P_c(t')dt'$$

Question 19
Mettre en oeuvre la méthode des trapèze pour calculer l'énergie totale consommée par le cycliste.

Profile de puissance record

Question 20
Proposer une méthode pour obtenir ce graphe en tenant compte des données gps issues du fichier gpx.

Question 21
A partir de l'estimation de la puissance instantanée précédente déterminer le PPR.

Exercice 147 – (FIC-006)

Vous trouverez sur le site de classe un fichier `cats.csv`. Il contient des mesures de corpulence de chats. Vous y trouverez trois colonnes : le sexe de chaque chat (colonne `Sex`), son poids en kilogrammes (colonne `Bwt`) et le poids de son cœur en grammes (colonne `Hwt`).

Écrire un script `Python` permettant de lire ce fichier et d'écrire dans un autre fichier, pour chaque sexe, le poids moyen et le poids du cœur moyen.

Exercice 148 – (FIC-007)

Vous trouverez sur le site de classe un fichier `sainte_lyon.csv`. Il contient la liste des 31 premiers concurrents de la course de trail nocture **La Saintélyon** de l'édition 2017 reliant Saint-Etienne à Lyon. Vous y trouverez deux colonnes : le nom de chaque concurrent (colonne `Nom`) et le temps mis pour parcourir le parcours de 72 km (colonne `Temps`) au format $heures:minutes:secondes$.

Question 1 *Écrire un script* `Python` *permettant de lire ce fichier et d'écrire dans un autre fichier, pour chaque nom, le temps en secondes et la vitesse en km/h.*

Exercice 149 – (FIC-008)

Le fichier sur lequel vous allez travailler a été chiffré en utilisant le chiffre de César. C'est un code par décalage très simple, qui fonctionne avec une clef $c \in [\![0,26[\![$. En numérotant les lettres de 0 (lettre `a`) à 25 (lettre `z`), chaque lettre de numéro r est remplacée par la lettre de numéro $(r+c)\%26$.

Par exemple, si la clef est 2 (lettre `c`), le `a` est transformé en `c`, le `b` en `d`, ..., le `x` en `z`, le `y` en `a` et le `z` en `b`.

Indication : dans cet exercice, vous pourrez introduire `alphabet="abcdefghijklmnopqrstuvwxyz"`. Ainsi, si $i \in [\![0,26[\![$, `alphabet[`i`]` sera la lettre n^o i.

Question 1
Appliquez la tranformation de clef 24 (lettre `y`) à votre texte. Quelle est le $(\alpha+50)^e$ mot de ce nouveau texte (on numérote à partir de zéro) ?

Pour déchiffrer un texte chiffré avec la clef c, il suffit d'appliquer la transformation de clef $(26-c)\%26$. Cette dernière clef sera appelée *clef de déchiffrement* du texte. Par exemple, si la clef de chiffrement est 3 (lettre `d`), alors la clef de déchiffrement sera 23 (lettre `x`).

La cryptanalyse du chiffre de César, c'est-à-dire l'obtention de la clef à partir du texte chiffré, peut se faire par analyse de fréquences.

La fréquence d'une lettre dans un texte est le rapport entre le nombre d'occurences dans ce texte de cette lettre et le nombre total de lettres du texte.

Question 2
Quelle est la fréquence de la lettre `e` dans votre texte chiffré ?

Pour comparer deux tableaux de fréquences de lettres $t=[t_0,\dots,t_{25}]$ et $u=[u_0,\dots,u_{25}]$, on utilise la distance

$$d(t,u)=\sum_{k=0}^{25}(t_k-u_k)^2.$$

Le fichier `frequences.txt` contient une liste de fréquences des lettres, que l'on supposera être celle du français. Vous le retrouverez dans le tableau **??**.

Question 3
Quelle est la distance entre le tableau des fréquences des lettres de votre texte chiffré et celui des fréquences données dans le fichier `frequences.txt` ?

L'algorithme de déchiffrement est le suivant : pour chaque clef $c \in [\![0,26[\![$, on applique le code de César de clef c au texte (ce qui donne un second texte), puis l'on calcule le tableau des fréquences des lettres de ce second texte, et enfin l'on calcule la distance entre ce tableau et celui donné par le fichier `frequences.txt`.

On sélectionne alors la clef qui minimise les distances calculées ci-dessus.

Question 4
Quelle est la clef de déchiffrement de votre texte ?

Question 5
Quelle est le $(\alpha+100)^{\text{e}}$ mot du texte déchiffré (on numérote à partir de zéro) ?

Exercice 150 – (FIC-009)

Copier et coller le fichier "dipole_electrique.txt" que vous trouverez sur le site de la classe dans votre répertoire personnel. Celui-ci contient des données numériques, obtenues en TP d'électronique, correspondant à différentes mesures de l'intensité traversant un dipôle et de la tension entre ses bornes. On souhaite récupérer ces données et les exploiter.

Ouvrir le fichier "dipole_electrique.txt" à l'aide d'un éditeur de texte (de type bloc-notes) et observer son contenu.

Question 1
Écrire un programme Python que vous appellerez `lit_dipole` et qui prendra en argument une variable de type chaine de caractère qui sera le nom du fichier de données. Cette fonction renverra deux listes notées texttt{I} et texttt{U}, l'une contenant les intensités mesurées, et l'autre contenant les tensions mesurées.

Question 2
Créer une fonction que l'on notera `tracer_dipole` qui prendra en argument deux listes représentant respectivement l'intensité I et U. Cette fonction permettra de représenter graphiquement les couples de points (I,U). On sauvegardera la figure avec le nom 'tp06_noms_q02.png'

Question 3
Conjecturer alors une relation reliant la tension aux bornes du dipôle à l'intensité le traversant.

Exercice 151 – (FIC-010)

Le fichier `body.csv` contient des données anatomiques de 507 personnes adultes. Pour chaque personne, plusieurs données sont disponibles :

- identifiant de la personne (nombre entre 1 et 507) ;
- 21 mesures de la corpulence de la personne (en centimètres) ;
- âge (en années) ;
- poids (en kilogrammes) ;
- taille (en centimètres) ;
- sexe (1 : homme et 0 : femme).

On pensera d'abord à ouvrir ce fichier dans un éditeur de texte puis dans un tableur afin de bien visualiser ces données.

L'indice de masse corporelle (IMC) est le rapport entre le poids d'un individu (exprimé en kg) et le carré de sa taille (exprimée en m). C'est un indicateur (simpliste) permettant de mesurer le sous/sur-poids d'un individu adulte. Ainsi, on (enfin, l'OMS) considère que si l'IMC d'un adulte est :

- dans $]0;18,5[$, la personne est en sous-poids ;
- dans $[18,5;25[$, la personne a un poids normal ;
- dans $[25;30[$, la personne est en sur-poids ;
- dans $[30;+\infty[$, la personne est obèse.

Quelques rappels sur les chaînes. Il existe un moyen de « découper » des chaînes de caractères en `Python` : c'est la méthode `split`. Réciproquement, la méthode `join(t)` appliquée à une chaîne `x` permet de concaténer toutes les chaînes du tableau `t`, séparées par `x`. Il existe aussi des outils de conversion de nombres flottants en chaînes de caractère, et vice-versa. Tout cela s'utilise comme suit.

```
>>> s = '123,45,2;1587,45,;45'
>>> s.split(',')
>>> s.split(';')
>>> sep = '<'
>>> t = ['GA','BU','ZO','MEU']
>>> sep.join(t)
>>> str(123.456)
>>> float('456.123')
```

Enfin, on voudra bien entendu représenter les données contenues dans `body.csv` sous forme de *tableau à double entrée*, c'est-à-dire comme une matrice. On représentera le tableau sous forme de liste `Python`, ligne par ligne, chaque ligne étant elle même une liste `Python`.

▪ **Exemple** Pour représenter le tableau $T = \begin{pmatrix} a & b & c \\ 4 & 5 & 6 \end{pmatrix}$, on fera comme suit.

```
>>> T = [['a','b','c'],['4','5','6']]
>>> T
>>> T[1]
>>> T[0][1]
>>> T.append(['x','y','z'])
>>> T
```

▪

Question 1
Écrire une fonction `lecture(nom_de_fichier)` qui prend en argument une chaîne de caractères `nom_de_fichier` contenant le chemin du fichier `body.csv` et qui renvoie un tableau à double entrées, où :

- la ligne 0 contient le titre de chaque colonne ;
- si $1 \leq i \leq 507$, chaque ligne i contient successivement :
  - l'identifiant i ;
  - les 21 mesures de corpulence pour l'individu n$^{\text{o}}$ i ;
  - l'âge de l'individu n$^{\text{o}}$ i ;
  - le poids de l'individu n$^{\text{o}}$ i ;
  - la taille de l'individu n$^{\text{o}}$ i ;
  - le sexe de l'individu n$^{\text{o}}$ i.

Indication : on choisira à chaque fois le type le plus convenable pour chaque donnée.

Question 2
Écrire une fonction `calcul_imc(T)` prenant en argument un tableau à double entrées que l'on supposera être celui renvoyé par la fonction `lecture` et qui renvoie un

tableau à double entrées S possédant 26 colonnes, identique à T sur ses 25 premières colonnes et dont la dernière colonne contient l'IMC de chaque individu.

Question 3

Écrire une fonction `catimc(nom_de_fichier)` qui prend en argument une chaîne de caractères `nom_de_fichier` contenant le chemin du fichier `body.csv` et qui renverra une liste Python de longueur 4 qui, pour chaque catégorie de poids, comptera le nombre de personnes du jeu de données dans cette catégorie.

Ainsi, le premier élément du tableau renvoyé par `catimc(nom_de_fichier)` sera le nombre d'individus en sous-poids dans `body.csv`.

Question 4

Écrire une fonction `ecrire(T,nom_de_fichier,sep=',')` qui écrit le contenu d'un tableau T analogue à celui obtenu en sortie de la fonction `calcul_imc()` (voir question 151) dans le fichier `nom_de_fichier`.

On veillera notamment à séparer les colonnes par le séparateur `sep`.

Vous enverrez le fichier produit à l'enseignant. Les instructions de rendu sont données dans le préambule.

Exercice 152 – (FIC-011)

Ouvrir le fichier `complot_contre_lamerique.txt` dans python (on prendra soin de nommer la variable contenant cet objet). Ce fichier contient un extrait du livre de Philip Roth, Complot contre l'Amérique.

Question 1

Que fait chacune des méthodes textttread(), textttreadline() et textttreadlines()? Quels sont les types des valeurs que chacune des ces fonctions renvoient?

Question 2

Que représentent les symboles `\t` et `\n`?

Question 3

Écrire une fonction Python `carac(nom_de_fichier)` qui renvoie un tableau contenant le nombre de caractères de chaque ligne du fichier `nom_de_fichier`, en excluant les symboles `\t` et `\n`.

Question 4

Écrire une fonction Python `somme_carac(nom_de_fichier)` qui renvoie le nombre de caractères total contenu dans le fichier `nom_de_fichier` , en excluant les symboles `\t` et `\n`.

Indication : attention au type de `nom_de_fichier`!

Question 5

Écrire une fonction Python `compte_carac(carac,nom_de_fichier)` qui renvoie pour un caractère de l'alphabet (noté `carac`) son nombre d'occurrences contenu dans le fichier `nom_de_fichier` sans tenir compte de la casse.

Question 6

Écrire une fonction Python `stat_carac(nom_de_fichier)` qui renvoie une liste de 26 éléments donnant le nombre d'occurrences de chaque lettre de l'alphabet contenu dans le texte `nom_de_fichier` sans tenir compte de la casse.

Indication :

On pourra créer une variable globale contenant tous les caractères de l'alphabet :

`alphabet='abcdefghijklmnopqrstuvwxyz'`.

Question 7

Écrire une fonction Python `tracer_occurrences(carac,nom_de_fichier)` qui trace en fonction du numéro de la lettre dans l'alphabet son nombre d'occurrences dans le fichier `nom_de_fichier`. La figure obtenue avec l'extrait du livre complot contre l'Amérique de Philip Roth sera sauvegardée sous le nom `tp06_vosnoms_q10.png`.

Question 8

Écrire une fonction Python `tracer_stat_occurrences(nom_de_fi` qui trace en fonction de chaque lettre dans l'alphabet sa fréquence d'apparition dans le fichier `nom_de_fichier` en pourcentage par rapport au nombre total de caractères présents en excluant les symboles `\t` et `\n`. Le tracé devra avoir l'apparence d'un histogramme. La figure obtenue avec l'extrait du livre complot contre l'Amérique de Philip Roth sera sauvegardée sous le nom `tp06_vosnoms_q11.png`.

Indication :

- Pour faire apparaître les lettres en abscisses, on pourra utiliser la fonction `xticks` du module `matplotlib.pyplot`.
- Pour tracer les barres l'histogramme on pourra utiliser tout simplement des segments verticaux.

On donne le fichier `frequence_wikipedia.csv` téléchargeable sur le site de la classe qui contient les fréquences en pourcentage d'apparition des 26 lettres de l'alphabet dans tous les articles français de Wikipedia. On pourra l'ouvrir avec un éditeur de texte pour observer son contenu.

Question 9

Écrire une fonction Python `stat_carac_wikipedia(nom_de_fichi` qui à partir du fichier `nom_de_fichier` contenant les statistiques de fréquences de caractères (sous le même format que dans le fichier `frequence_wikipedia.csv`) renvoie une liste donnant en fonction de la position de la lettre dans l'alphabet sa fréquence en %.

Question 10

Écrire une fonction Python `tracer_stat_wikipedia()` qui trace en fonction de chaque lettre dans l'alphabet, sa fréquence d'apparition dans tous les articles français de wikipedia en pourcentage par rapport au nombre total de caractères présents. Le tracé devra avoir l'apparence d'un histogramme. La figure obtenue sera sauvegardée sous le nom `tp06_vosnoms_q13.png`.

Question 11

Utiliser les différentes fonctions réalisées précédemment pour comparer les fréquences de caractères entre l'extrait de complot contre l'Amérique de Philip Roth et celles issues de l'ensemble des articles français de wikipedia. Conclure.

Exercice 153 – (FIC-012)

Le fichier `embrunman2019.csv` contient les résultats de l'ultratriatlon de Embrun 2019. Les données concernent dans l'ordre :

- le classement ;
- le numéro de dossard ;
-

On pensera d'abord à ouvrir ce fichier dans un éditeur de texte puis dans un tableur afin de bien visualiser ces données.

Quelques rappels sur les chaînes. Il existe un moyen de « découper » des chaînes de caractères en Python : c'est la méthode `split()`. Réciproquement, la méthode `join(t)`

appliquée à une chaîne `x` permet de concaténer toutes les chaînes du tableau `t`, séparées par `x`. Il existe aussi des outils de conversion de nombres flottants en chaînes de caractère, et vice-versa. Tout cela s'utilise comme suit.

```
>>> s = '123,45,2;1587,45,;45'
>>> s.split(',')
>>> s.split(';')
>>> sep = '<'
>>> t = ['GA','BU','ZO','MEU']
>>> sep.join(t)
>>> str(123.456)
>>> float('456.123')
```

Enfin, on voudra bien entendu représenter les données contenues dans `embrunman2019.csv` sous forme de *tableau à double entrée*, c'est-à-dire comme une matrice. On représentera le tableau sous forme de liste `Python`, ligne par ligne, chaque ligne étant elle même une liste `Python`.

6 Tracer de courbes

Exercice 154 – (PLT-001)

Une fonction pseudo-périodique est une fonction qui permet de décrire des phénomènes physiques du type «oscillatoire-amortis». Cette fonction peut-être le produit d'une fonction harmonique sinusoïdale et d'une fonction exponentielle décroissante, de la forme :

$$f : x \to \sin(x) \cdot \exp(\alpha \cdot x)$$

Le coefficient α doit être un réel négatif strictement pour avoir un régime amorti. On prendra dans la suite $\alpha = -0,1$.

On souhaite obtenir le tracé ci-dessous (voir figure **??**), faisant apparaître la courbe représentative de la fonction f sur le segment $[0, 6\pi]$, mais également les courbes enveloppes $(x \mapsto \pm\exp(\alpha \cdot x))$ et la courbe représentative de la fonction harmonique non amortie $(x \mapsto \sin x)$.

Question 1
Donner les instructions permettant de réaliser ce tracé. On veillera à bien préciser les modules utilisés et à utiliser les instructions permettant d'obtenir le tracé complet.

Exercice 155 – (PLT-002)

Question 1 *Tracer la courbe de la fonction* Arctan *sur l'intervalle* $[-10, 10]$.

Question 2 *Tracer les courbes des fonctions* $x \mapsto x$, $x \mapsto x^2$ *et* $x \mapsto \sqrt{x}$ *sur l'intervalle* $[0, 1]$ *et sur un même graphique.*

Question 3 *Tracer en échelle logarithmique les courbes* $x \mapsto x$, $x \mapsto x^2$ *et* $x \mapsto \sqrt{x}$ *sur* $[1, 10^5]$ *et sur un même graphique. Que lit-on sur les pentes des courbes obtenues ?*

Exercice 156 – (PLT-003)

Le code suivant permet de tracer les graphes des fonction arccos et arcsin, dans la figure **??**.

```
import matplotlib.pyplot as plt
import math as m
from numpy import linspace

x1 = linspace(-1,1,200)
x2 = linspace(0,m.pi,200)
arccos = [m.acos(t) for t in x1]
cos = [m.cos(t) for t in x2]

plt.clf()
plt.plot(x1,arccos,color='b',label='$\\arccos(t)
    $', linewidth=2)
plt.plot(x2,cos,color='r',label='$\\cos(t)$',
    linewidth=2)
plt.plot([-0.5*m.pi,m.pi],[-0.5*m.pi,m.pi],
    color = 'g',
  linestyle = '--', label = 'Première␣
      bissectrice␣du␣plan')
plt.xlabel('$t$')
plt.legend(loc=0)
plt.axes()
plt.savefig('ex_avance.png')
```

Exercice 157 – (PLT-004)

En utilisant `Python` on peut tracer de nombreux types de graphiques. Nous allons utiliser une bibliothèque regroupant de (très) nombreuses fonctions de tracé : `matplotlib`. En fait, cette bibliothèque est bien trop vaste, nous n'utiliserons que sa sous-bibliothèque `matplotlib.pyplot`.

Commencez par ouvrir votre IDE, puis créez un script nommé `tp05_ex_sin.py`. Recopiez dedans le script suivant.

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

n = 20
x = [k*10/n for k in range(n)]
y = [sin(t) for t in x]

plt.clf()
plt.plot(x,y,label='sin(x)')
plt.xlabel('x')
plt.legend(loc=0)
plt.title('Tracé␣du␣sinus␣sur␣[0,10]')
plt.savefig('tp05_ex_sin.png')
```

Éxécutez ce script et vérifiez que la figure créée est en tout point semblable à celle présente sur le site de classe.

Question 1
Quel est le type de `x` ?

Question 2
Comment `Python` représente-t-il graphiquement une fonction ?

Question 3
Où le tracé s'arrête-t-il ? Pourquoi ?

On rappelle que l'on peut simplement créer une liste d'abscisses en utilisant la fonction `linspace` de la bibliothèque de calcul `numpy`.

Chargez cette fonction dans l'interpréteur interactif par la commande

```
from numpy import linspace
```

puis consultez son manuel par la commande `help(linspace)`.

Question 4
Écrire une fonction `ex_sin(nom_de_fichier)` permettant tracer de manière plus appropriée la courbe de l'exemple précédent et qui enregistre l'image produite dans le fichier `nom_de_fichier`. Vous produirez alors une image, que vous enverrez à votre enseignant.

Indication : Attention au type de `nom_de_fichier` ! Notamment, on supposera que l'extension du fichier est déjà présente dans `nom_de_fichier`.

Question 5 <
Écrire une fonction `transitoire(A,nom_de_fichier)`

qui enregistre dans le fichier `nom_de_fichier` le graphe des fonctions

$$t \mapsto A\left(1-\mathrm{e}^{-\frac{t}{\tau}}\right)$$

sur $[0,10]$, pour chaque $\tau \in \left\{\frac{1}{2};1;2;4;8\right\}$. Vous produirez une image (vous choisirez A), que vous enverrez à votre enseignant.

Indication : Attention au type de `nom_de_fichier` !

Exercice 158 – (PLT-005)

On s'intéresse maintenant à l'approximation d'un signal périodique par des fonctions trigonométriques. On considère sur $\mathbb{R}$ la fonction créneau, impaire et périodique de période 2, définie par $C(1)=0$ et sur $]0,1[$ par

$$C : t \mapsto 1.$$

Soit aussi la fonction triangle, définie sur $\mathbb{R}$, paire et périodique de période 2, définie sur $[0,1]$ par

$$T : t \mapsto 1-2t.$$

Ces deux fonctions sont représentées sur la figure **??**.

On peut montrer que, en tout réel t où C est continue,

$$C(t)=\frac{4}{\pi}\sum_{p=0}^{+\infty}\frac{1}{2p+1}\sin((2p+1)\pi t).$$

De même, pour tout $t\in\mathbb{R}$,

$$T(t)=\frac{8}{\pi^2}\sum_{p=0}^{+\infty}\frac{1}{(2p+1)^2}\cos((2p+1)\pi t).$$

Ce symbole $\sum_{p=0}^{+\infty}$ doit être vu comme une limite et sera défini dans le cours de mathématiques. On approche alors, pour tout $t\in\mathbb{R}$, $C(t)$ et $T(t)$ respectivement par les sommes

$$C_n(t)=\frac{4}{\pi}\sum_{p=0}^{n}\frac{1}{2p+1}\sin((2p+1)\pi t)$$

et

$$T_n(t)=\frac{8}{\pi^2}\sum_{p=0}^{n}\frac{1}{(2p+1)^2}\cos((2p+1)\pi t).$$

Les physiciens appellent le terme en $p=0$ le *fondamental* et les autres termes les *harmoniques*.

Question 1
Écrire une fonction `creneau(t)` renvoyant la valeur de C(`t`) (pour `t` réel).

Indication : on pourra considérer la parité de la partie entière de `t`.

Question 2
Écrire une fonction `sp_creneau(n,t)` renvoyant la valeur de $C_{\mathtt{n}}$(`t`) (pour `n` entier et `t` réel).

Question 3
Écrire une fonction `fourier_creneau(nom_de_fichier)` ne renvoyant rien et enregistrant dans `nom_de_fichier` le graphe sur l'intervalle $[0,4]$ de C, celui de son fondamental (*i.e*, C_0), ainsi que ceux de C_3, C_5 et C_{100}. Vous produirez une image, que vous enverrez à votre enseignant.

Pour les plus rapides, voici des questions supplémentaires.

Question 4
Écrire une fonction `triangle(t)` renvoyant la valeur de T(`t`) (pour `t` réel).

Question 5
Écrire une fonction `sp_triangle(n,t)` renvoyant la valeur de $T_{\mathtt{n}}$(`t`) (pour `n` entier et `t` réel).

Question 6
Écrire une fonction `fourier_triangle(nom_de_fichier)` ne renvoyant rien et enregistrant dans `nom_de_fichier` le graphe sur l'intervalle $[0,4]$ de T, celui de son fondamental (*i.e*, T_0), ainsi que ceux de T_3, T_5 et T_{100}. Vous produirez une image, que vous enverrez à votre enseignant.

7 Architecture

Exercice 159 – (ARCHI-000)

Cette séance commencera par une présentation du fonctionnement d'un ordinateur (relative à la fin du chapitre 1 vu en cours). La seconde partie de la séance sera consacrée au TP en lui-même (Partie 9) : vous démonterez les anciens ordinateurs du lycée pour repérer et étudier leurs composants.

1. **Lisez attentivement tout l'énoncé avant de commencer.**
2. Après la séance, vous devez rédiger un compte-rendu de TP et l'envoyer au format électronique à votre enseignant.
3. Le seul format accepté pour l'envoi d'un texte de compte-rendu est le format PDF. Votre fichier s'appellera impérativement `tp01_durif_pessoles.pdf`, où « `durif` » et « `pessoles` » sont à remplacer par les noms des membres du binôme.
4. Ce TP est à faire en binôme, vous ne rendrez donc qu'un compte-rendu pour deux.
5. Pensez à vous munir d'un appareil photo (un smartphone est suffisant).
6. Avant d'envoyer le compte rendu, merci de vérifier que les photos incluses dedans sont d'un poids raisonnable.

8 Généralités sur le fonctionnement d'un ordinateur.

▪ **Exemple** **Exemple ou contre-exemple d'ordinateur au sens de la définition apportée dans le cours ?**

1. Automobile
2. Thermostat d'ambiance (mécanique)
3. Thermostat d'ambiance (électronique)
4. Téléphone portable (smartphone)
5. PC de bureau
6. Lecteur MP3
7. Box de votre FAI
8. Métier Jacquard

▪

9 Démontage d'un PC et étude de ses composants.

Avant tout, notez bien qu'on ne démonte pas les ordinateurs de la salle de TP mais ceux fournis par l'enseignant dans ce but !

Deuxième chose : on ne touche pas l'intérieur du PC, ou alors le moins possible et en ayant pris soin au préalable de se décharger de son électricité statique en touchant la carcasse du PC.

Chaque binôme dispose d'un PC. Le but du jeu est de l'ouvrir, d'identifier les différents composants (où est le disque dur ? la carte mère ? le processeur ? la mémoire vive ?), puis de le refermer, le tout sans casse.

N'oubliez pas de prendre des photos de l'intérieur pour votre compte-rendu.

Vous préciserez sur le compte-rendu les différents éléments que vous avez identifiés, de préférence à partir de photos.

Attention : vous ne démonterez pas ni la carte mère, ni le bloc d'alimentation électrique (vous pourrez ôter ce dernier du l'ordinateur).

Exercice 160 – (ARCHI-001)

1. **Lisez attentivement tout l'énoncé avant de commencer.**
2. Après la séance, vous devez rédiger un compte-rendu de TP et l'envoyer au format électronique à votre enseignant.
3. Le seul format accepté pour l'envoi d'un texte de compte-rendu est le format PDF. Votre fichier s'appellera impérativement `tp02_kleim_durif.pdf`, où « `kleim` » et « `durif` » sont à remplacer par les noms des membres du binôme.
4. Ce TP est à faire en binôme, vous ne rendrez donc qu'un compte-rendu pour deux.
5. Vous préciserez en préambule de votre compte-rendu les noms des membres du binôme ainsi que le système d'exploitation sur lequel vous avez travaillé.
6. Ayez toujours un crayon et un papier sous la main. Quand vous réfléchissez à une question, utilisez-les !
7. Vous devez être autonome. Ainsi, avant de poser une question à l'enseignant, merci de commencer par :
   - relire l'énoncé du TP (beaucoup de réponses se trouvent dedans) ;
   - relire les passages du cours [6] relatifs à votre problème ;
   - effectuer une recherche dans l'aide disponible sur votre ordinateur (ou sur internet) concernant votre question.

   Il est alors raisonnable d'appeler votre enseignant pour lui demander des explications ou une confirmation !

Le but de ce TP est de vous faire de manipuler un système d'exploitation par lignes de commandes, puis de prendre en main `Python`.

Attention : les étudiants travaillant sous Unix (Linux et MacOS) répondront d'abord aux questions des parties 9.2 et 9.3, ceux travaillant sous Windows répondront d'abord aux questions des parties 9.4 et 9.5. Tous les étudiants finiront par répondre aux questions de la partie **??**.

Pour les manipulations de fichier, vous utiliserez le code source du logiciel `Python`, que vous pourrez trouver dans le dossiers groupes. à l'adresse suivante (où `X` est à remplacer par `1` ou `2`) :

`~/groupes/mpsX/données/TP02/cpython-4243df51fe43`

6. Dans le cas fort improbable où vous ne vous en souviendriez pas.

9.1 Introduction : terminal et shell

Unix a été inventé à un moment où l'utilisateur avait la possibilité d'interagir avec l'ordinateur via un *terminal*, c'est-à-dire la combinaison d'un clavier et d'un écran pouvant écrire (en général) 25 lignes de 80 caractères (en une seule couleur, généralement vert ou orange sur fond noir). Cette façon d'interagir avec la machine peut paraître archaïque de nos jours mais elle est pourtant d'une puissance diabolique.

Vous trouverez un émulateur de terminal dans le menu « Applications », sous-menu « Accessoires », et sélectionnez « LXTerminal ». Ceci démarre un programme, appelé *shell* ou *interprète de commandes*. Ce *shell* vous donne quelques informations, et affiche un symbole `$`, appelé invite (ou *prompt* en anglais) signe qu'il attend vos ordres.

Pour lui donner un ordre, il suffit de taper le nom de la commande désirée, éventuellement suivie d'un espace puis d'options ou d'arguments séparés par des espaces, puis de valider par la touche Entrée.

Sous Windows, le programme `cmd.exe` permet d'ouvrir une console fonctionnant sous le même principe. Une remarque : les noms de dossiers comportant des espaces devront être écrits entre guillemets (par exemple : `"Mes Documents"`). De manière générale, c'est une très mauvaise idée de placer des espaces dans des noms de dossiers (ou de fichiers).

Vous trouverez dans la table **??** les équivalents Windows de quelques commandes Unix.

9.2 Unix : utilisation d'un terminal.

Une commande très pratique est `man` : elle permet d'obtenir le manuel de quasiment toutes les commandes. On l'utilise sous la forme `man` *page* où *page* est la page de manuel désirée.

Question 1
Tapez `man less`. Que se passe t-il ?

Vous pouvez faire défiler le texte ligne par ligne avec Entrée ou page par page avec la barre d'espace et quitter `man` avec la touche q.

Question 2
Quelle commande permet d'afficher la page suivante dans `less` ? de quitter `less` ?

9.3 Unix : fichiers et répertoires

Sous Unix (dont la distribution GNU/Linux est un représentant) les fichiers sont organisés hiérarchiquement en une arborescence unique de répertoires. La racine de cette arborescence, c'est-à-dire le répertoire supérieur de la hiérarchie contenant tous les fichiers auxquels à accès le système, est noté `/`. Ses sous-répertoires directs (de l'ordre de la dizaine ou quelques dizaines de répertoires), comme `home`, `media`, ... sont notés `/home`, `/media`, ...

Le *chemin absolu* d'un fichier est l'adresse complète de son emplacement, débutant de la racine et passant par tous les sous-répertoires requis pour atteindre le fichier visé.

Le *chemin relatif* d'un fichier est l'adresse de son emplacement, écrite à partir d'un emplacement de l'arborescence que l'on appelle *répertoire courant* (en anglais : *current working directory*). Ce répertoire courant est initialisé par défaut à un point prédéterminé de l'arborescence (répertoire « maison », ou *home*), mais peut ensuite être modifié.

Question 3
Que fait précisément la commande `ls` ?

La commande `cd` permet de changer de répertoire courant, `pwd` permet d'afficher le répertoire courant. En particulier, la commande `cd` *d*,où *d* est le nom absolu ou relatif d'un répertoire, change le répertoire courant en *d*. Essayez avec `cd /usr/bin` par exemple.

Question 4
Que fait `cd` sans argument (*i.e.* `cd` non suivi du nom d'un répertoire) ?

Question 5
Changer le répertoire courant (par exemple en `/usr/bin`). Que fait `cd ~` ? De quoi ~ est-il l'abréviation ? Quel est le chemin absolu du répertoire ~ ?

Question 6
Que fait la commande `mkdir` ? En utilisant une ligne de commande, créer dans le répertoire ~ un sous-répertoire nommé `TP02`. Qu'observe-t-on en exécutant la commande `ls ~` ?

Question 7
Après avoir éxécuter la commande `cd TP02` à partir du répertoire ~, qu'affiche la commande `pwd` ?

Copier le dossier `cpython-4243df51fe43` depuis le dossier `groupes/mpsX/TP02` dans le répertoire `~/TP02` en passant par un explorateur de fichiers. Exécuter la commande `cd` dans le terminal.

Pour Unix : dans un répertoire, les fichiers et répertoires dont le nom commence par un point sont dits *cachés*.

Question 8
En consultant le manuel de `ls`, trouver la commande qui permet d'afficher les fichiers et répertoires cachés.

Question 9
Dans `~/TP02`, vous pouvez alors voir deux répertoires cachés. Quels sont leurs noms ?

En fait, dans chaque répertoire du système, il existe deux répertoires cachés avec ces deux mêmes noms.

Question 10
Que désignent les répertoires `.` et `..` ? Que donne un `cd` sur chacun de ces répertoires ?

Question 11
Changez le répertoire courant en `~/TP02/cpython-4243df51fe43/L`
Que fait alors `cd ../../multiprocessing` ?

Question 12
Comment obtenir grâce à la commande `ls` et l'option `-l` la taille de tous les fichiers de `TP02/cpython-4243df51fe43/Modules`, en les triant par ordre croissant de taille ?

Question 13
Facultatif : Avec la commande précédente, que remarquez-vous quant à la taille des sous-répertoires de `TP02/cpython-4243df51fe43/Modules` ?

Question 14
Facultatif : En utilisant la commande `du`, donner la taille du répertoire `cjkcodecs`. En comparant ce résultat à celui de la question précédente, que pouvez-vous dire de la manière dont Linux considère les répertoires ?

9.4 Windows : utilisation d'un terminal.

Une commande très pratique est `help` : elle permet d'obtenir le manuel de quasiment toutes les commandes. On l'utilise sous la forme `help` *page* où *page* est la page de manuel désirée.

Question 15
Tapez `help more`. Que se passe t-il ?

Vous pouvez faire défiler le texte ligne par ligne avec Entrée ou page par page avec la barre d'espace et quitter `man` avec la touche q.

Question 16
Quelle commande permet d'afficher la page suivante dans `more` ? de quitter `more` ?

9.5 Windows : fichiers et répertoires

Sous Windows, les fichiers sont organisés hiérarchiquement en plusieurs arborescences de répertoires. La racine de chacune de ces arborescences, c'est-à-dire le répertoire supérieur de la hiérarchie contenant tous les fichiers auxquels à accès le système, est nommé par une lettre (`A:`, `B:` *etc.*).

Le séparateur de nom de dossier est alors `\`. Par exemple, le sous-répertoire `Home` de la racine `C:` est noté `C:\Home`.

Le *chemin absolu* d'un fichier est l'adresse complète de son emplacement, débutant de la racine et passant par tous les sous-répertoires requis pour atteindre le fichier visé.

Le *chemin relatif* d'un fichier est l'adresse de son emplacement, écrite à partir d'un emplacement de l'arborescence que l'on appelle *répertoire courant* (en anglais : *current working directory*). Ce répertoire courant est initialisé par défaut à un point prédéterminé de l'arborescence (répertoire « maison », ou *home*), mais peut ensuite être modifié.

Question 17
Que fait précisément la commande `dir` ?

La commande `cd`, avec un argument, permet de changer de répertoire courant. Sans argument, elle permet d'afficher le répertoire courant. En particulier, la commande `cd` *d*,où *d* est le nom absolu ou relatif d'un répertoire, change le répertoire courant en *d*. Essayez avec par exemple.

Question 18
Changer le répertoire courant. Que fait `cd %UserProfile%` ? De quoi `%UserProfile%` est-il l'abréviation ? Quel est le chemin absolu du répertoire `%UserProfile%` ?

Question 19
Que fait la commande `mkdir` ? En utilisant une ligne de commande, créer dans le répertoire `%UserProfile%` un sous-répertoire nommé `TP02`. Qu'observe-t-on en exécutant la commande `dir %UserProfile%` ?

Question 20
Après avoir éxécuter la commande `cd TP02` à partir du répertoire ~, qu'affiche la commande `cd` ?

Copier le dossier `cpython-4243df51fe43` depuis le dossier `groupes/mpsX/TP02` dans le répertoire `%UserProfile%\TP02` en passant par un explorateur de fichiers.

Pour Unix : dans un répertoire, les fichiers et répertoires dont le nom commence par un point sont dits *cachés*. Ce n'est pas la même chose sous Windows.

Question 21
En consultant le manuel de `dir`, trouver la commande qui permet d'afficher les fichiers et répertoires cachés.

Question 22
Faire un inventaire des fichiers de `%UserProfile%\TP02` ? Y en a-t-il des cachés ?

Question 23
Que désignent les répertoires `.` et `..` ? Que donne un `cd` sur chacun de ces répertoires ?

Question 24
Changez le répertoire courant en `%UserProfile%\TP02\cpython-42`
Que fait alors `cd ..\..\multiprocessing` ?

Question 25
Comment obtenir grâce à la commande `dir` et l'option `-l` la taille de tous les fichiers de `TP02\cpython-4243df51fe43\Modules`, en les triant par ordre croissant de taille ?

Exercice 161 – (ARCHI-002)

Lancer PYZO ou IDLE. Un *interpréteur de commandes*, ou `shell`, s'affiche. Le symbole `>>>` signifie que `Python` attend vos instructions.

Sitôt une instruction tapée et validée (par la touche « Entrée »), le `shell` effectue le calcul demandé puis affiche un résultat, ou un message d'erreur. Il est extrêmement important de bien lire ces messages d'erreur, et de les comprendre !

Question 1
Taper dans le *shell* les instructions suivantes.

```
x = 42
y = 42.
type(x)
type(y)
x = x+y
x
type(x)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Question 2
Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples numériques.

`+` `-` `*` `**` `/` `//` `%`

Question 3
Taper dans le *shell* les instructions suivantes.

```
B = 42 > 41.
type(B)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Question 4
Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples numériques.

```
==    !=    <    >    <=    >=
```

Question 5
Taper dans le *shell* les instructions suivantes.

```
3/0 > 5
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Question 6
Taper dans le *shell* les instructions suivantes.

```
B = (42 > 41) or ( 3/0 > 5).
type(B)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Question 7
Décrire ce que font les opérations suivantes, après les avoir étudiées sur des exemples logiques.

```
or    and    not
```

Question 8
Taper dans le *shell* les instructions suivantes.

```
x = -3
abs(x)
help(abs)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Question 9
Taper dans le *shell* les instructions suivantes.

```
import math as m
import numpy as np
m.sin(m.pi)
np.sin(np.pi)
np.sin([0,np.pi])
m.sin([0,m.pi])
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Question 10
Taper dans le *shell* les instructions suivantes.

```
L = [0,1,2,3,4,5,6]
type(L)
L[0]
L[6]
L[-1]
L[-2]
L[7]
L[1:4]
L[2:8]
L.append(7)
L
L = L.append(8)
L
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Nous avons vu comment utiliser des fonctions et des bibliothèques. Nous pouvons bien entendu créer nos propres fonctions (et bibliothèques).

Dans PYZO ou IDLE, ouvrir un nouveau fichier (CTRL+N ou File / New file). L'enregistrer (CTRL + S ou File / Save) sous le nom `TP02.py`.

Question 11
Taper dans cette fenêtre le script suivant.

```
"""TP n02"""
def somme(n) :
    """Renvoie 0 + 1 + 2 + ... + n
    Precondition : n entier naturel"""
    return n*(n+1) // 2
```

Enregistrer puis appuyer sur la touche Crtl+MAJ+E (sous PYZO) ou F5 (sous IDLE). Le *shell* doit s'afficher. Taper dans le *shell* les instructions suivantes.

```
somme(42)
somme(42.)
somme(-1515)
help(somme)
```

Que se passe-t-il ? Qu'est-ce que cela signifie ?

Question 12
Comment peut-on utiliser la fonction écrite précédemment dans un *autre* script `Python` ?

Exercice 162 – (ARCHI-004)

Question 1 *Rappeler les trois fonctions qui permettent de définir un ordinateur.*

Question 2 *Parmi les propositions suivantes, décrire quelles sont celles présentes dans l'architecture de Von Neumann*

(a) Mémoire virtuelle ;
(b) Processus ;
(c) Processeur ;
(d) Canal de communication ;
(e) Mémoire vive ;
(f) Mémoire de masse.

Question 3 *Donner la signification du mot OS. Décrire brièvement à quoi ça sert et donner 3 exemples.*

Modéliser

10 Représentation des nombres

Exercice 163 – (NBR-000)

Question 1
Trouver la somme de tous les entiers qui sont la somme des factorielles de leurs chiffres, en écriture décimale. On écrira une fonction `enigme()` renvoyant le résultat demandé.

Exercice 164 – (NBR-001)

Question 1
Que renvoie la fonction suivante ? Pourquoi ?

```
def mystere():
    a,b,n = 1,1.,0
    while a==b:
        a,b,n = 2*a,2*b,n+1
    return n
```

Exercice 165 – (NBR-002)

Question 1
Que renvoie la fonction suivante ? Pourquoi ?

```
def mystere():
    a,b,n = 1,1.,0
    while a==b:
        a,b,n = 2*a+1,2*b+1,n+1
    return n
```

Exercice 166 – (NBR-003)

En mathématiques, la méthode de la dichotomie à partir d'un segment $[a_0, b_0]$ donne deux suites infinies de réels (a_n) et (b_n) vérifiant $\forall n \in \mathbb{N}$, $a_n < b_n$.

Question 1 *Est-ce le cas en informatique ?*

Question 2 *Justifier la réponse et l'illustrer par un script.*

Question 3 *Donner une estimation du plus petit nombre dénormalisé que l'on peut représenter sur 64 bits.*

Question 4 *Conclure.*

Exercice 167 – (NBR-004)

Un diviseur strict d'un entier naturel n est un diviseur positif de n différent de n. Un nombre est déficient s'il est strictement plus grand que la somme de ses diviseurs stricts.

Par exemple, 15 a pour diviseurs stricts 1, 3 et 5. Comme $1+3+5<15$, 15 est déficient.

Question 1
Calculer le nombre de nombres déficients de 10^6 (inclu) à 10^6+10^5 (exclu).

Exercice 168 – (NBR-005)

On admet que pour tout entier $n \geq 1$ il existe une unique suite $a_1, \ldots, a_p$ vérifiant :

- $n = \sum_{k=1}^{p} a_k \times k!$
- $a_p \neq 0$ et $\forall k \in [\![1, p]\!]$, $0 \leq a_k \leq k$.

On appellera *écriture en base factorielle* de l'entier n la chaîne $a_1 - a_2 - \cdots - a_p$.

Par exemple, $42 = 0\times 1! + 0\times 2! + 3\times 3! + 1\times 4!$, l'écriture de 42 en base factorielle est donc $0-0-3-1$.

Question 1
Donner l'écriture en base factorielle de $\alpha^3+\alpha^2+\alpha+10^5$.

11 Intégration numérique

Exercice 169 – (INT-001)

Question 1

Tracer les nuages de points correspondant, pour tout $0 \leq p < 1000$ aux calculs de $\int_0^1 t^p \mathrm{d}t$ par les méthodes suivantes : rectangles à gauche, rectangles à droite, trapèzes, fonction `quad` de `scipy` et la valeur exacte. On utilisera 1000 rectangles ou trapèzes à chaque fois.

Question 2

Produire un second graphique en faisant apparaître les erreurs d'approximations des quatre premières méthodes en utilisant une échelle appropriée.

Question 3

Tracer les nuages de points correspondant, pour tout $1 \leq n < 1000$ aux calculs de $\int_0^1 t^2 \mathrm{d}t$ par les méthodes suivantes : rectangles à gauche, rectangles à droite, trapèzes et la valeur exacte, en utilisant n rectangles ou trapèzes.

Question 4

Produire un second graphique en faisant apparaître les erreurs d'approximations des quatre premières méthodes en utilisant une échelle appropriée.

Exercice 170 – (INT-002)

Le but est d'obtenir un encadrement de $I = 4\int_0^1 \frac{\mathrm{d}x}{1+x^2}$

Question 1

Compléter cet algorithme et le coder en python afin d'obtenir une valeur approchée de I par la méthode des rectangles à gauche en utilisant les champs suivants : x+h | (b-a)/n | h | somme+f(x) | f(a).

Question 2

A partir de la question précédente, écrire une fonction que l'on appellera `rect_gauche`, qui prendra en argument une fonction `f`, les variables `a` et `b` définissant le domaine d'intégration et le paramètre `n` définissant le nombre de subdivisions. Cette fonction renverra la valeur approché de `I`.

```
a=0
b=1
n=100
h=
x=a
somme=
for k in range(1,n) :
    x=
    somme=
print(somme*
```

Question 3

Modifier cet algorithme pour que la méthode soit celle des rectangles à droite. On appellera la fonction associée `rect_droit`, elle prendra les mêmes arguments et en renverra la valeur approché de `I`.

Question 4

Définir la fonction `f(x)` permettant d'utiliser les deux fonctions précédentes avec la fonction à intégrer.

Question 5

Tester ces fonctions en augmentant le nombre de subdivisions. et en traçant les résultats obtenus sur l'estimation de I pour différentes valeurs de n. Vous sauvegarderez le graphe obtenu sous le nom **"tp10_q05_vos_noms.png"** et vous l'enverrez à votre professeur (On veillera à choisir des valeurs de n et une échelle de représentation graphique pertinentes).

Question 6

Justifier que la méthode des rectangles à droite donne un minorant de I et que la méthode des rectangles à gauche donne un majorant.

Pour tout $n \in \mathbb{N}$ soit l'intégrale :

$$I_n = \int_0^1 \frac{x}{1+x^n}\mathrm{d}x$$

Question 7

Écrire une fonction que l'on appellera `calcul_in` d'argument n qui renvoie une valeur approchée de I_n par une la méthodes des rectangles à gauche (avec une subdivision de 100 intervalles).

Question 8

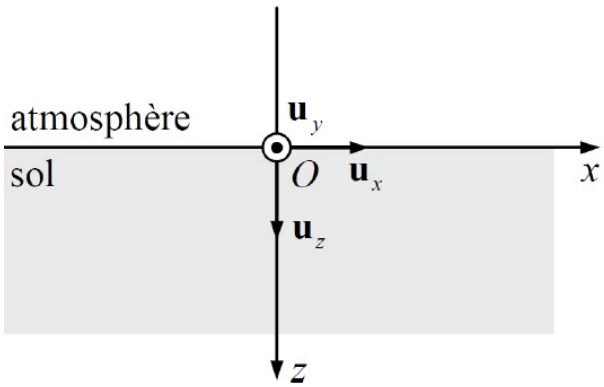
A l'aide d'un graphe choisi judicieusement (que vous sauvegarderez sous le nom **"tp10_q08_vos_noms.png"** et vous l'enverrez à votre professeur), afficher quelques valeurs de cette suite afin de conjecturer sa monotonie et son comportement asymptotique.

Exercice 171 – (INT-003)

La température dans le sol terrestre étant initialement constante, égale à 5°C , on cherche à déterminer à quelle profondeur minimale il est nécessaire d'enterrer une canalisation d'eau pour qu'une brusque chute de la température de sa surface à -15°C n'entraine pas le gel de cette canalisation après 10 jours.

Les hypothèses sont les suivantes :

- la température en un point quelconque du sol et de sa surface à tout instant $t < 0$ est constante et égale à $T_0 = 278\ K$ ($\theta_0 = 5^o C$) ;
- la température à la surface du sol, confondue avec le plan d'équation $z = 0$, passe brutalement à l'instant $t = 0$, de $T_0 = 278\ K$ à $T_1 = 258\ K$ ($\theta_1 = -15^o C$) et se maintient à cette valeur pendant t_f =10 jours.



On peut montrer que la température $T(z,t)$ à la profondeur z et à l'instant t est donnée par la relation suivante :

$$T(z,t) = T_1 + (T_0 - T_1)\mathrm{erf}\left(\frac{z}{2\sqrt{Dt}}\right)$$

où erf(x) désigne la fonction définie par :

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} \mathrm{d}u$$

Données numériques : $D = 2,8 \cdot 10^{-7}\ m^2 \cdot s^{-1}$ (diffusivité thermique du sol terrestre).

Question 1

Écrire une fonction Python, appelée trapeze, prenant en argument une fonction `f`, les variables `a` et `b` définissant le domaine d'intégration et le paramètre `n` définissant le nombre de subdivisions.

Question 2

Écrire une fonction Python, appelée erf, prenant en paramètre un nombre réel positif ou nul x, un entier nb correspondant au nombre de subdivisions de la méthode d'intégration et retournant la valeur de erf(x).

Question 3

Écrire une fonction Python, appelée Temperature, prenant en paramètre la profondeur z (exprimée en m) et le temps t (exprimé en s) et retournant la valeur de la température $T(z,t)$ (On pourra utiliser 500 subdivisions pour les calculs d'intégration).

Question 4

Écrire un programme Python permettant de créer une liste, nommée ListeErreur, contenant les valeurs de la fonction erf(x) pour x variant par pas de 0,05 dans l'intervalle [0;2] (On pourra utiliser 500 subdivisions pour les calculs d'intégration).

Question 5

En déduire, à 1 cm près, à quelle profondeur minimale z_{min} il est nécessaire d'enterrer une canalisation d'eau pour qu'une brusque chute de la température de la surface du sol de 5°C à -15 °C n'entraine pas le gel de cette canalisation au bout de 10 jours.

Exercice 172 – (INT-004)

On démontre que la longueur L de la courbe $y = x^2$ pour $x \in [0;\alpha]$ dans un repère orthonormal est donnée par : $L = \int_0^\alpha \sqrt{1+4x^2}\mathrm{d}x$.

Question 1

Calculer L à 10^{-3} près par la méthode des rectangles à gauche.

Question 2

Calculer L à 10^{-3} près par la méthode des rectangles à droite.

Question 3

Calculer L à 10^{-3} près par la méthode des trapèzes.

Exercice 173 – (INT-periodependule)

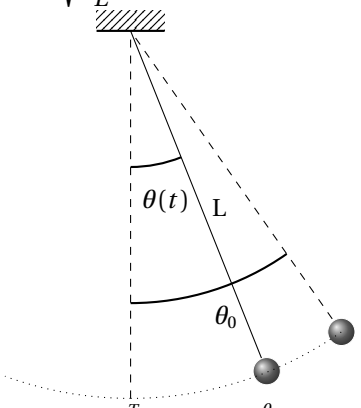
On considère un pendule de masse $m = 1+0,01 \cdot \alpha$, de longueur $L = \alpha$ que l'on abandonne, sans vitesse initiale, à un angle de $\theta_0 = \dfrac{\pi}{4 \cdot \alpha}$ avec $g = 9,81\,\mathrm{m\,s^{-2}}$.

En appliquant la conservation de l'énergie, on trouve l'équation suivante :

$$\frac{1}{2} m L^2 \dot{\theta}(t)^2 + m g l (1 - \cos\theta(t)) = m g L (1 - \cos\theta_0)$$

On en déduit : $\dfrac{\mathrm{d}\theta(t)}{\mathrm{d}t} = \sqrt{\dfrac{2g}{L}(\cos\theta(t) - \cos\theta_0)}$.

Ce qui donne : $\mathrm{d}t = \dfrac{\mathrm{d}\theta(t)}{\sqrt{\dfrac{2g}{L}(\cos\theta(t) - \cos\theta_0)}}$.



La période est donc : $T = \displaystyle\int_0^T \mathrm{d}t = 4 \times \int_0^{\theta_0} \frac{\mathrm{d}\theta(t)}{\sqrt{\dfrac{2g}{L}(\cos\theta(t) - \cos\theta_0)}}$.

On rappelle que la période avec l'approximation des petites oscillations est donnée par : $T_0 = \sqrt{\dfrac{L}{g}} \times 2\pi$.

R $T = \displaystyle\int_0^{\theta_0} f(\theta)\mathrm{d}\theta$ est une intégrale généralisée car $f(\theta)$ tend vers l'infini. Donc certaines méthodes numériques d'intégration donnent des résultats infinis.

Pour les méthodes posant problèmes, il faut alors remplacer la borne supérieur θ_0 par $\theta_0(1-\varepsilon)$ avec $\varepsilon = 10^{-3}$.

Question 1

Calculer T_0.

Pour les méthodes d'intégration numérique, on renverra les résultats avec 100 subdivisions.

Question 2

Calculer T (noté T_g) par la méthode des rectangles à gauche.

Question 3

Calculer T (noté T_d) par la méthode des rectangles à droite.

Question 4

Calculer T (noté T_t) par la méthode des trapèzes.

On note $\varepsilon_g = |T_g - T_0|$, $\varepsilon_d = |T_d - T_0|$, $\varepsilon_t = |T_t - T_0|$, les erreurs entre les différentes approximation et l'estimation de T_0.

Question 5

Renvoyer ε_g, ε_d, ε_t.

12 Équations différentielles

Exercice 174 – (EQD-000)

On considère une équation différentielle $(\mathscr{E})$: $y' = F(y,t)$, où F est une fonction de $\mathbb{R}^n \times \mathbb{R}$ dans $\mathbb{R}^n$, et où l'inconnue y est une fonction de $\mathscr{C}^1(\mathbb{R},\mathbb{R}^n)$, avec la condition initiale $y(t_0) = y_0$.

Question 1

Décrire le principe de la méthode d'Euler. On donnera clairement la relation de récurrence qui est au coeur de cette méthode, en expliquant bien ce que représente la suite vérifiant cette relation de récurrence.

Question 2

Écrire en `Python` une fonction mettant en œuvre la méthode d'Euler et permettant de résoudre numériquement $(\mathscr{E})$ sur le segment $[a,b]$, où $a,b \in \mathbb{R}$, $a < b$. Vous écrirez une docstring décrivant tous les arguments de cette fonction.

On considère l'équation différentielle d'inconnue $y \in \mathscr{C}^2(\mathbb{R})$ et les conditions initiales suivantes :

$$y'' + \cos(t)y' - t^2 y = \mathrm{e}^t, \qquad y(0) = 4,\ y'(0) = 2. \quad (\mathscr{E})$$

Question 3

Donnez une fonction F et une variable X telle que $(\mathscr{E})$ soit équivalente à l'équation $X' = F(X,t)$. On précisera bien les ensembles de définition et d'arrivée de F, et l'ensemble auquel appartient la variable X, ainsi que la condition initiale à utiliser.

Exercice 175 – (EQD-001)

Autour de la dynamique gravitationnelle

Présentation

Modéliser les interactions physiques entre un grand nombre de constituants mène à l'écriture de systèmes différentiels pour lesquels, en dehors de quelques situations particulières, il n'existe aucune solution analytique. Les problèmes de dynamique gravitationnelle et de dynamique moléculaire en sont deux exemples. Afin d'analyser le comportement temporel de tels systèmes, l'informatique peut apporter une aide substantielle en permettant leur simulation numérique. L'objet de ce sujet est l'étude de solutions algorithmiques en vue de simuler une dynamique gravitationnelle afin, par exemple, de prédire une éclipse ou le passage d'une comète.

Quelques fonctions utilitaires

Question 1

Donner la valeur des expressions python suivantes :

- $[1,2,3]+[4,5,6]$
- $2*[1,2,3]$

Question 2

Écrire une fonction python $smul$ à deux paramètres, un nombre et une liste de nombres, qui multiple chaque élément de la liste par le nombre et renvoie une deuxième liste sans modifier la première. Par exemple, *smul(2, [1, 2, 3])* renvera *[2, 4, 6]*.

Question 3

Déterminer la complexité de cet algorithme en fonction de la taille de la liste.

Étude de schémas numériques

Soient y une fonction de classe $\mathscr{C}^2$ sur $\mathbb{R}$ et t_{min} et t_{max} deux réels tels que $t_{min} < t_{max}$. On note I l'intervalle $[t_{min}, t_{max}]$. On s'intéresse à une équation différentielle du second ordre de la forme :

$$\forall t \in I \qquad y''(t) = f(y(t)), \tag{3}$$

où f est une fonction donnée, continue sur $\mathbb{R}$. De nombreux systèmes physiques peuvent être décrits par une équation de ce type.

On suppose connues les valeurs $y_0 = y(t_{min})$ et $z_0 = y'(t_{min})$.

Mise en forme du problème

Pour résoudre numériquement l'équation différentielle (3), on introduit la fonction $z : I \to \mathbb{R}$ définie par

$$\forall t \in I, z(t) = y'(t). \tag{4}$$

On considère l'équation :

$$Y' = F(Y,t) \tag{5}$$

Question 4

Pour quelle variable Y et quelle fonction F, l'équation (3) peut se mettre sous la forme de l'équation (5) ?

Soit n un entier strictement supérieur à 1 et $J_n = [\![0, n-1]\!]$. On pose $h = \dfrac{t_{max} - t_{min}}{n-1}$ et $\forall i \in J_n$, $t_i = t_{min} + i \cdot h$. On peut montrer que, pour tout entier $i \in [\![0, n-2]\!]$,

$$Y(t_{i+1}) = Y(t_i) + \int_{t_i}^{t_{i+1}} Y'(t)\,dt \tag{6}$$

La suite du problème exploite les notations introduites dans cette partie et présente deux méthodes numériques dans lesquelles l'intégrale précédente est remplacée par une valeur approchée.

Schéma d'Euler explicite

Dans le schéma d'Euler explicite, chaque terme sous le signe intégrale est remplacé par sa valeur prise en la borne inférieure.

Question 5

Ecrire une fonction $euler(F, tmin, tmax, Y_0, n)$ qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i\in J_n}$, $(z_i)_{i\in J_n}$ ainsi que la liste du temps $(t_i)_{i\in J_n}$.

Pour illustrer cette méthode, on considère l'équation différentielle,

$$\forall t \in I, \quad y''(t) = -\omega^2 y(t) \tag{7}$$

dans laquelle ω est un nombre réel.

Question 6

Expliciter en langage `Python` la fonction $F(Y,t)$ et la fonc-

Modéliser

tion $f(y)$ qui permettront de mettre en oeuvre ce problème.

13.1

La mise en oeuvre de la méthode d'Euler explicite génère le résultat graphique donné figure **??**. Dans un système d'unités adapté, les calculs ont été menés en prenant $y_0 = 3$, $z_0 = 0$, $t_{min} = 0$, $t_{max} = 3$, $\omega = 2\pi$ et $n = 100$.

Question 7
Écrire la suite d'instructions permettant à partir de la fonction F et après avoir défini la fonction $euler$ de la mettre en oeuvre et produire le tracé de la figure **??**.

Schéma de Verlet

Le physicien français Loup Verlet a proposé en 1967 un schéma numérique d'intégration d'une équation de la forme (3) dans lequel, en notant $f_i = f(y_i)$ et $f_{i+1} = f(y_{i+1})$, les relations de récurrence s'écrivent, pour tout entier $i \in [\![0, n-2]\!]$:

$$y_{i+1} = y_i + hz_i + \frac{h^2}{2} f_i \quad \text{et} \quad z_{i+1} = z_i + \frac{h}{2}(f_i + f_{i+1}). \tag{8}$$

Question 8
Écrire une fonction $verlet(f, tmin, tmax, Y_0, n)$ qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i\in J_n}$ et $(z_i)_{i\in J_n}$, ainsi que la liste du temps $(t_i)_{i\in J_n}$.

13.2

On reprend l'exemple de l'oscillateur harmonique défini par l'équation (7) et on compare les résultats obtenus à l'aide des schémas d'Euler et de Verlet.

Question 9
Écrire la suite d'instructions permettant à partir de la fonction f et après avoir défini la fonction $verlet$ de la mettre en oeuvre et produire le tracé de la figure **??**.

Comparaison qualitative des schémas numériques.

On rappelle que l'énergie pour un tel oscillateur est proportionnelle à :

$$\mathscr{E} = \frac{1}{2}(y')^2 + \frac{\omega^2}{2} y^2.$$

Question 10
Conclure sur la stabilité des deux schémas numériques.

Exercice 176 – (EQD-002)

13 Etude d'un trafic routier

Ce sujet concerne la conception d'un logiciel d'étude de trafic routier. On modélise le déplacement d'un ensemble de voitures sur des files à sens unique (voir Figures **??** et **??**). C'est un schéma simple qui peut permettre de comprendre l'apparition d'embouteillages et de concevoir des solutions pour fluidifier le trafic.

13.3

Préliminaires

Dans un premier temps, on considère le cas d'une seule file, illustré par la Figure **??**. Une file de longueur n est représentée par n cases. Une case peut contenir au plus une voiture. Les voitures présentes dans une file circulent toutes dans la même direction (sens des indices croissants, désigné par les flèches sur la Figure **??**) et sont indifférenciées.

Question 1
Expliquer comment représenter une file de voitures à l'aide d'une liste de booléens.

Question 2
Donner une ou plusieurs instructions `Python` permettant de définir une liste A représentant la file de voitures illustrée par la Figure **??**.

Question 3
Soit L une liste représentant une file de longueur n et i un entier tel que $0 \leqslant i < n$. Définir en Python la fonction `occupe(L, i)` qui renvoie `True` lorsque la case d'indice i de la file est occupée par une voiture et `False` sinon.

Question 4
Combien existe-t-il de files différentes de longueur n ? Justifier votre réponse.

Question 5
Écrire une fonction `egal(L1, L2)` retournant un booléen permettant de savoir si deux listes $L1$ et $L2$ sont égales.

Question 6
Que peut-on nom_de_fichier de la complexité de cette fonction ?

Base de données

On modélise ici un réseau routier par un ensemble de croisements et de voies reliant ces croisements. Les voies partent d'un croisement et arrivent à un autre croisement. Ainsi, pour modéliser une route à double sens, on utilise deux voies circulant en sens opposés. La base de données du réseau routier est constituée des relations suivantes :

- Croisement(id, longitude, latitude)
- Voie(id, longueur, id_croisement_debut, id_croisement_fin)

Dans la suite on considère c l'identifiant (id) d'un croisement donné.

Question 7
Écrire la requête SQL qui renvoie les identifiants des croisements atteignables en utilisant une seule voie à partir du croisement ayant l'identifiant c.

Question 8
Écrire la requête SQL qui renvoie les longitudes et latitudes des croisements atteignables en utilisant une seule voie, à partir du croisement c.

Question 9
Que renvoie la requête SQL suivante ?

Simulation dynamique

On introduit les bibliothèques `Math` et `NumPy` à l'aide des lignes suivantes :

```
import math as m
import numpy as np
```

On s'intéresse maintenant à l'apparition des voitures en début de file. On se place alors dans le cas d'un parking lors d'une sortie d'usine entre 17 h et 19 h. Afin de simplifier le problème on considère qu'à $t=0$, il est 17 h et à $t=1$, il est 19 h.

Tous les employés de l'usine ne sortent pas au même moment. On donne sur la Figure **??** l'évolution de $Q(t)$ qui représente le nombre de véhicules quittant le stationnement par unité de temps, pour $t \in [0,1]$. On considère que cette fonction est nulle en dehors de cet intervalle. On appelle `db_stat_max` le maximum de Q entre 0 et 1.

$$\forall t \in [0,1],\ Q(t) = \texttt{db\_stat\_max} \cdot \mathrm{e}^{\left(1 - \frac{1}{4t(1-t)}\right)}$$

Question 10
Écrire une fonction `Q(t)` permettant d'obtenir, pour tout réel t, le nombre de véhicules quittant le stationnement par unité de temps. On considère `db_stat_max` comme une variable globale. Faire attention aux cas où `t = 0` et `t = 1`.

Question 11
Écrire une fonction `integrale(f, a, b, n)` permettant, avec la méthode des trapèzes, d'estimer l'intégrale de `f` entre `a` et `b` à partir de `n` points équirépartis.

Question 12
On appelle $N(t)$ le nombre de véhicules ayant quitté le stationnement jusqu'à l'instant t. Écrire une fonction `Nb(t, n)` renvoyant une valeur approchée de ce nombre, en utilisant n points.

La suite de cette partie permettant d'obtenir $N(t)$ par une autre méthode, la fonction `Nb` ne sera plus utilisée.

La vitesse de sortie des véhicules est limitée par le nombre de véhicules sortants. En effet, quand peu de véhicules sortent du parking, ces derniers peuvent circuler à vitesse maximale V_{max}. En revanche, lorsque beaucoup de véhicules sont en mouvement, ces derniers se ralentissent les uns les autres. On appelle K la concentration de véhicules cherchant à rejoindre la sortie du parking à un instant donné et K_{sat} le nombre de véhicules à partir duquel la vitesse de sortie est minimale (V_{min}).

L'expression de la vitesse des véhicules V en fonction de $K \in [0, K_{sat}]$ est donnée par

$$V(K) = \frac{V_{max} - V_{min}}{2} \cdot \left(1 + \cos\left(\pi \cdot \frac{K}{K_{sat}}\right)\right) + V_{min}.$$

Question 13
Écrire une fonction `V(K)` prenant en entrée un flottant `K` et renvoyant la valeur de la vitesse correspondante. On considère `Vmin`, `Vmax` et `Ksat` comme des variables globales.

La conservation du nombre total de véhicules conduit au système différentiel suivant :

$$\frac{\mathrm{d}N}{\mathrm{d}t}(t) = Q(t) \tag{9}$$
$$\frac{\mathrm{d}K}{\mathrm{d}t}(t) = Q(t) - K(t)\cdot V(K) \tag{10}$$
$$\frac{\mathrm{d}S}{\mathrm{d}t}(t) = K(t)\cdot V(K) \tag{11}$$

où $S(t)$ désigne le nombre de véhicules sortis du parking à l'instant t.

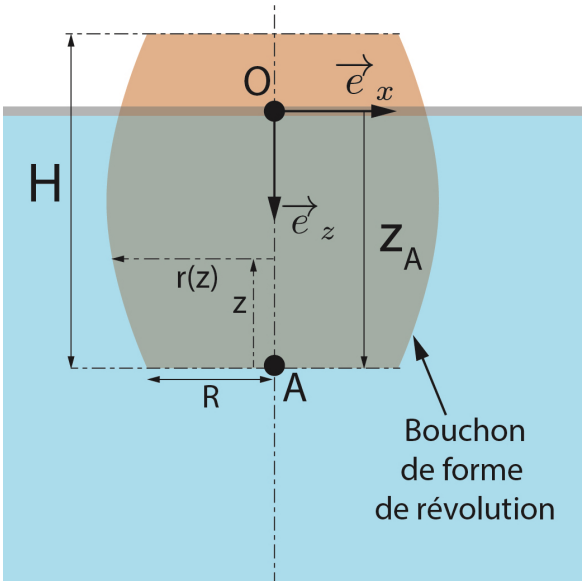
Question 14
Rappeler la relation de récurrence de la méthode d'Euler explicite pour un problème défini par son équation différentielle $y'(t) = F(y(t), t)$ avec $h = \Delta t$ le pas de temps supposé constant.

On considère que cette méthode est implémentée dans une fonction `odeint(F, Y0, T)` où `F` est la fonction associée au problème de Cauchy, `Y0` est un tableau `NumPy` contenant les valeurs initiales de la fonction Y et `T` un tableau `NumPy` contenant les différents pas de temps.

Question 16
Commenter en quelques lignes les allures des courbes de la Figure **??**.

Exercice 177 – (EQD-003)

Ce sujet concerne la dynamique d'un bouchon en liège flottant dans un verre d'eau (voir figure suivante).



On note :
- R le rayon de la base du bouchon ;
- ρ_e la masse volumique de l'eau ;
- ρ_b la masse volumique du bouchon ;
- $z_A(t)$ la position verticale du bas du bouchon selon la direction $\overrightarrow{e}_z$ qui dépend du temps t par rapport à O (point situé sur la surface de l'eau) ;
- H la hauteur du bouchon.

Le bouchon possède une symétrie de révolution. Ainsi, son rayon, noté $r(z)$, dépend de la coordonnée z de la manière suivante :

$$\begin{array}{rccl} r : & [0,H] & \to & \mathbb{R}, \\ & z & \mapsto & R \cdot \left[1 + 0{,}1 \cdot \sin\left(\pi \cdot \frac{z}{H}\right)\right]. \end{array}$$

Le volume immergé du bouchon dépend de la position z_A et est donné par la relation suivante :

Modéliser

$$V_i(z_A)=\begin{cases} 0 & \text{si} \quad z_A<0\,; \\ \pi\cdot\displaystyle\int_{z=0}^{z_A}(r(z))^2\,\mathrm{d}z & \text{si} \quad 0\le z_A\le H\,; \\ \pi\cdot\displaystyle\int_{z=0}^{H}(r(z))^2\,\mathrm{d}z & \text{si} \quad z_A>H. \end{cases}$$

On note V le volume total du bouchon correspondant à $V_i(H)$.

Le théorème de la résultante dynamique appliqué au bouchon selon la direction $\overrightarrow{e}_z$ donne :

$$M\cdot\frac{\mathrm{d}^2z_A}{\mathrm{d}t^2}(t)=P-F_p(z_A(t))+F_v\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right). \quad (\mathscr{E})$$

Les grandeurs suivantes interviennent dans l'équation $(\mathscr{E})$.

- $M=\rho_b\cdot V$, la masse du bouchon (égale au produit de sa masse volumique avec son volume).
- $P=\rho_b\cdot V\cdot g$, le poids du bouchon.
- g, l'accélération de la pesanteur.
- F_p, la force de poussée. Elle est égale au produit de la masse volumique de l'eau (ρ_e) avec g et du volume immergé ($V_i(z_A(t))$), *i.e.*

$$F_p(z_A(t))=\rho_e\cdot g\cdot V_i(z_A(t)).$$

- $F_v\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)$ est la force de frottement visqueux. Elle est donnée par la formule

$$F_v\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)=-\frac{1}{2}\rho_e\cdot C_x\cdot S\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)\cdot\left(\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)^2$$

- $S\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)$ est la surface apparente du bouchon vis-à-vis du fluide et est définie de la manière suivante.
  - Si le bouchon est totalement hors de l'eau, cette surface est nulle.
  - Si le bouchon descend et est (au moins partiellement) immergé :
    - si $0\le z<H/2$, $S\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)=\pi\cdot r(z)^2$,
    - si $z\ge H/2$, $S\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)=\pi\cdot r(H/2)^2$.
  - Si le bouchon remonte et est (au moins partiellement) immergé :
    - si $0\le z<H/2$, $S\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)=0$.
    - si $H/2\le z\le H$, la surface à prendre en compte est celle de la couronne : $S\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)=-\pi\cdot\left(r(H/2)^2-r(z)^2\right)$
    - si $z>H$, $S\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)=-\pi\cdot r(H/2)^2$.

  La force de frottement s'opposant au mouvement, si le bouchon remonte, la résultante de cette force selon la direction $\overrightarrow{e}_z$ est positive, d'où le signe $-$ placé ici.
- C_x est le coefficient de trainée aérodynamique.

Dans toute la suite de ce devoir, on pourra supposer que les grandeurs suivantes (et uniquement celles-ci) ont été définies.

```
R = 1E-2 # m, rayon minimum du bouchon de ré
    volution
H = 4.5*1E-2 # m, hauteur du bouchon
rho_eau = 1000 # kg / m**3 masse volumique de l
    'eau
rho_b = 240 # kg / m**3, masse volumique du liè
    ge
g = 9.81 # m / s**2, accélération de la
    pesanteur en
Cx = 1 # Coefficient de trainée aérodynamique
N = 1000 # Nombre de trapèzes pour les calculs
    d'intégrales
```

Question 1

Écrire la fonction `T(f,a,b,N)` permettant de donner l'estimation de $\displaystyle\int_{z=a}^{b}f(z)\mathrm{d}z$ par la méthode des trapèzes, avec N trapèzes sur le segment $[a,b]$.

Question 2

Écrire une fonction `volume_immerge(z)` qui renvoie le volume immergé du bouchon en fonction de la profondeur du bas du bouchon (noté z) en utilisant la fonction définie à la question précédente.

Question 3

Écrire l'instruction permettant de calculer le volume total du bouchon, que l'on affectera à la variable `V`.

Question 4

Écrire une fonction `Fv(z,zp)` qui renvoie la force de frottement visqueux $F_v\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)$.

Question 5

Exprimer $\frac{\mathrm{d}^2z_A}{\mathrm{d}t^2}(t)$ en fonction de ρ_e, ρ_b, $V_i(z_A(t))$, $F_v\left(z_A(t),\frac{\mathrm{d}z_A}{\mathrm{d}t}(t)\right)$ et $z_A(t)$.

On souhaite résoudre cette équation différentielle (équation $\mathscr{E}$) avec pour conditions initiales :

$$\begin{cases} z_A(t=0)=-0,2 \\ \dfrac{\mathrm{d}z_A}{\mathrm{d}t}(t=0)=0 \end{cases} \quad (\text{CI})$$

Question 6

Définir l'expression de $Z(t)$, Z_0 et de $F(Z(t),t)$ pour que l'équation différentielle $(\mathscr{E})$ avec les conditions initiales (CI) soit équivalente au problème de Cauchy d'ordre 1 :

$$Z'(t)=F(Z(t),t) \quad \text{et} \quad Z(0)=Z_0. \quad (\mathscr{F})$$

Écrire une suite d'instructions permettant de définir une telle fonction `F(Z,t)` ainsi que la condition initiale `Z0`.

Question 7

Écrire une fonction `euler(F, tmin, tmax, Z0, h)`

prenant en argument la fonction `F` définie précédemment, `tmin` et `tmax` définissant l'intervalle de résolution, le vecteur `Z0` définissant les consitions initiales ainsi que `h` le pas de discrétisation temporelle et permettant de résoudre de manière approchée l'équation $(\mathscr{F})$ par la méthode d'Euler.

14

On donne sur la figure **??** le résultat de l'application de la méthode d'euler pour simuler le comportement de la chute libre d'un bouchon à partir de 20 cm au dessus du niveau de l'eau.

Question 8 15

Commenter brièvement la cohérence physique d'une telle courbe. Pouvez-vous fournir une explication et une solution au problème observé, tout en restant dans le cadre de la méthode d'Euler?

Exercice 178 – (EQD-004)

Question 1

Déterminer une solution approchée des équations suivantes, à chaque fois par la méthode d'Euler, avec `odeint` et, si possible, théoriquement.

1. $y' = \sin(y)$, $y(0) = \dfrac{\pi}{2}$, sur $[0,1]$.
2. $y' + 3t^2 y = 5t^2$, $y(0) = 0$, sur $[0,10]$.
3. $\dfrac{y'}{1+y^2} = 1$, $y(0) = 1$, sur $\left[0, \dfrac{\pi}{4}\right[$.
4. $y' + y^2 = t$, $y(0) = 1$, sur $[0,1]$.

Exercice 179 – (EQD-005)

Question 1

Vectorialiser les équations suivantes (*i.e.* donner une équation vectorielle d'ordre 1 équivalente, dont on détaillera la construction de la variable et la condition initiale).

1. $y'' + 2y' + y = 5$, $y(0) = 3$, $y'(0) = 3$.
2. $y'' + 5t\sin(y) = t^3$, $y(0) = 0$, $y'(0) = 1$.
3. $t^3 y''' + 9t^2 y'' + 18ty' + 6y = 5$, $y(1) = 2$, $y'(1) = 0$, $y''(1) = 0$, sur $\mathbb{R}_+^*$.

Question 2

Déterminer une solution approchée de chacune des équations précédentes, à chaque fois par la méthode d'Euler, avec `odeint` et, si possible, théoriquement. 15.1

Exercice 180 – (EQD-006)

Question 1

Problème de cinétique chimique : avec trois réactifs, A, B et C. A se transforme en B qui se transforme en C (réactions d'ordre 1). Les concentrations de A, B et C suivent donc les équations suivantes.

$$\frac{\mathrm{d}[A]}{\mathrm{d}t} = -\alpha[A]$$
$$\frac{\mathrm{d}[B]}{dt} = \alpha[A] - \beta[B]$$
$$\frac{\mathrm{d}[C]}{\mathrm{d}t} = \beta[B]$$

On supposera $\alpha = 1\mathrm{s}^{-1}$, $\beta = 10\mathrm{s}^{-1}$ et au temps $t = 0$, $[A] = 1\mathrm{mol/l}$, $[B] = [C] = 0$. On veut regarder l'évolution jusqu'à $t = 6\mathrm{s}$.

Question 2

Tracer l'évolution des concentrations des réactifs, par la méthode d'Euler et avec scipy.

Exercice 181 – (EQD-007)

Pour commencer

Avant toute chose, on prendra soin de recopier la fonction `euler(F,a,b,y0,h)` donnée dans le chapitre 11 du cours.

Oscillations libres d'un pendule

On considère les oscillations libres d'un pendule. L'angle que fait ce pendule avec la verticale au temps t sera noté $\theta(t)$ et l'on étudie les oscillations du pendule entre 0 et 10 secondes. L'équation vérifiée par la fonction θ est alors

$$\ddot{\theta} + \alpha\dot{\theta} + \omega_0^2 \sin\theta = 0, \qquad (\mathscr{P})$$

où

- α (proportionnel au coefficient de frottements) est exprimé en s^{-1} ;
- ω_0 (la pulsation propre) est exprimé en s^{-1} .

Si le coefficient de frottement est nul, l'équation devient

$$\ddot{\theta} + \omega_0^2 \sin\theta = 0. \qquad (\mathscr{P}_{sf})$$

Enfin, on sait qu'au voisinage de 0 on a $\sin\theta = \theta + o(\theta^2)$. En supposant que les oscillations sont petites, on approche alors la dernière équation par

$$\ddot{\theta} + \omega_0^2 \theta = 0. \qquad (\mathscr{P}_{po})$$

Question 1

Pour quelle variable Θ et quelles fonctions F, F_{sf} et F_{po} les équations respectives $(\mathscr{P})$, $(\mathscr{P}_{sf})$ et $(\mathscr{P}_{po})$ sont-elles équivalentes aux équations suivantes?

$$\dot{\Theta} = F(\Theta, t), \quad \dot{\Theta} = F_{sf}(\Theta, t), \quad \dot{\Theta} = F_{po}(\Theta, t).$$

Approximation des petites oscillations par la méthode d'Euler

On suppose les oscillations petites et sans frottements, on prendra donc les valeurs numériques suivantes : $\alpha = 0s^{-1}$, $\omega_0 = \sqrt{2\pi}s^{-1}$. On étudiera les solutions sur un intervalle de temps de 10 secondes et l'on supposera que $\dot{\theta}(0) = 0s^{-1}$. Notamment, une méthode numérique effectuée avec n segments correspondra à un pas de $\dfrac{10}{n}$ secondes.

Question 2

Résoudre littéralement l'équation $(\mathscr{P}_{po})$.

Question 3

Écrire une fonction `Fpo(Theta,t)` prenant en argument un vecteur `Theta` et un nombre `t` et renvoyant F_{po}(`Theta,t`).

Question 4

Écrire une fonction `trace_po(th0,n,nom_de_fichier)` enregistrant dans `nom_de_fichier` le tracé de la solution exacte de $(\mathscr{P}_{po})$ ($\theta(0) =$ `th0`, $\dot{\theta}(0) = 0s^{-1}$) ainsi que, sur le

Modéliser

même graphe, celui de la solution obtenue par la méthode d'Euler avec `n` segments.

Vous enverrez à l'enseignant la figure produite pour $\theta_0 = 10^{-1}$ et $n = 10^2$.

15.2 Facultatif : approximation des petites oscillations par la méthode de Runge-Kutta d'ordre 4

Question 5

Sur le modèle de la fonction `euler(F,a,b,y0,pas)`, écrire une fonction Python `rk4(F,a,b,y0,pas)` mettant en œuvre la méthode de Runge-Kutta d'ordre 4 pour la fonction `F`, sur le segment $[a,b]$, avec un pas `pas` et la condition initiale `y0`. Cette fonction renverra un couple de tableaux.

Question 6

Compléter la fonction `trace_po(th0,n,nom_de_fichier)` pour y superposer le tracé obtenu par la méthode de Runge-Kutta. Que remarque-t-on ?

15.3 Oscillations sans frottements

On suppose les oscillations sans frottements, mais pas forcément petites on prendra donc les valeurs numériques suivantes : $\alpha = 0s^{-1}$, $\omega_0 = \sqrt{2\pi}s^{-1}$. On étudiera les solutions sur un intervalle de temps de 10 secondes et l'on supposera que $\dot\theta(0) = 0s^{-1}$. Notamment, une méthode numérique effectuée avec n points correspondra à un pas de $\frac{10}{n}$ secondes.

Question 7

Écrire une fonction `Fsf(Theta,t)` prenant en argument un vecteur `Theta` et un nombre `t` et renvoyant F_{sf}(`Theta,t`).

On veut maintenant vérifier numériquent l'approximation des petites oscillations. On connait déjà la solution exacte de $(\mathscr{P}_{po})$, il ne nous reste plus qu'à obtenir une approximation de la solution de $(\mathscr{P}_{sf})$. 16

Question 8

Écrire une fonction `approx_po(th0,n,nom_de_fichier)` enregistrant dans `nom_de_fichier` le tracé de la solution exacte de $(\mathscr{P}_{po})$ ($\theta(0) =$ `th0`, $\dot\theta(0) = 0s$) ainsi que, sur le même graphe, celui de la solution de $(\mathscr{P}_{sf})$ obtenue par la méthode d'Euler avec `n` segments. Que remarque-t-on ?

Vous enverrez à l'enseignant la figure produite pour $\theta_0 = 1$ et $n = 10^4$.

15.4 Facultatif : période des oscillations.

En dehors de la situation des petites oscillations, le mouvement du pendule est toujours périodique, mais sa période n'est pas $T_0 = \frac{2\pi}{\omega_0}$. Plus exactement, cette période dépend de la position initiale du pendule (le calcul précis n'est pas à votre portée). Nous allons tracer cette période en fonction de la position initiale du pendule.

Nous allons donc estimer la période de ces oscillations à partir d'une approximation obtenue par une méthode numérique.

On appelle *pic* d'un tableau une élément de ce tableau strictement plus grand que ses deux voisins. On considérera que le premier et le dernier élément d'un tableau ne sont pas des pics de ce tableau. On appelle alors *période* d'un tableau la moyenne des distances entre deux pics consécutifs.

Question 9

Écrire une fonction `periode(L)` renvoyant la période du tableau `L`.

On estimera la période du pendule simple à partir de la période du tableau des valeurs successives de θ obtenues par la méthode de Runge-Kutta. Plus exactement, si la période de ce tableau est p, si ce tableau comporte m valeurs et si les oscillations ont été mesurées sur un intervalle de temps de longueur T, alors l'estimation de la pédiode du pendule simple sera

$$T \times \frac{p}{m}.$$

Question 10

Écrire une fonction `periode_pendule(n)` prenant en argument un entier `n` et renvoyant le tableau des estimations de la période du pendule, pour les conditions initiales $\dot\theta(0) = 0s^{-1}$ et $\theta(0) = \frac{k\pi}{2n}$, pour chaque $k \in [\![1,n]\!]$.

On réfléchira à l'intervalle de temps utilisé ainsi qu'à la discrétisation utilisée dans la méthode de Runge-Kutta.

Question 11

Écrire une fonction `trace_periode(n,nom_de_fichier)` prenant en argument un entier `n` et enregistrant dans `nom_de_fichier` le tracé des points obtenus à la question précédente. On placera en abscisse les $\theta(0) = \frac{k\pi}{2n}$ et en ordonnée les périodes estimées.

Vous enverrez à votre enseignant la figure produite pour $n = 100$.

Oscillations forcées d'un pendule avec frottements

On suppose les oscillations forcées, avec frottements. La variable θ suit alors l'équation différentielle suivante.

$$\ddot\theta + \alpha\dot\theta + \omega_0^2 \sin\theta = \cos(\omega t). \qquad (\mathscr{P}_f)$$

On prendra les valeurs numériques suivantes : $\alpha = 0,5s^{-1}$, $\omega_0 = \sqrt{2\pi}s^{-1}$ et $\omega = 1s^{-1}$. On étudiera les solutions sur un intervalle de temps de 10 secondes. Notamment, une méthode numérique effectuée avec n segments correspondra à un pas de $\frac{10}{n}$ secondes.

Question 12

Pour quelle variable Θ et quelle fonction F_f l'équation $(\mathscr{P}_f)$ est-elle équivalente à l'équation suivante ?

$$\dot\Theta = F_f(\Theta, t).$$

Question 13

Écrire une fonction `Ff(Theta,t)` prenant en argument un vecteur `Theta` et un nombre `t` et renvoyant F_f(`Theta,t`).

Question 14

Écrire une fonction `trace_trajectoire_f(th0,thp0,n,nom_de_f`

prenant en argument deux nombres `th0`, `thp0`, un entier `n` ainsi qu'une chaîne `nom_de_fichier` et enregistrant dans `nom_de_fichier` la courbe de la trajectoire du pendule (c'est-à-dire des points $(t,\theta(t))$, obtenue en appliquant la méthode d'Euler avec `n` segments à l'équation $(\mathscr{P}_f)$, avec les conditions initiales $\theta(0)=$ `th0` et $\dot{\theta}(0)=$ `thp0`.

Vous enverrez à l'enseignant les figures produites pour les conditions initiales $\theta(0)=1$ et $\dot{\theta}(0)=2$ et pour $n=10^2$.

Question 15

Écrire une fonction `trace_phase_f(th0,thp0,n,nom_de_fichier)` prenant en argument deux nombres `th0`, `thp0`, un entier `n` ainsi qu'une chaîne `nom_de_fichier` et enregistrant dans `nom_de_fichier` le portrait de phase du pendule (c'est-à-dire des points $(\theta(t),\dot{\theta}(t))$, obtenu en appliquant la méthode d'Euler avec `n` segments à l'équation $(\mathscr{P}_f)$, avec les conditions initiales $\theta(0)=$ `th0` et $\dot{\theta}(0)=$ `thp0`.

Vous enverrez à l'enseignant les figures produites pour les conditions initiales $\theta(0)=1$ et $\dot{\theta}(0)=2$ et pour $n=10^2$.

Question 16

Écrire une fonction `odeint_f(th0,thp0,n)` prenant en argument deux nombres `th0` et `thp0` ainsi qu'un entier `n` et renvoyant le couple de listes `t_list,y_list` obtenu en appliquant la fonction `odeint` avec `n` segments à l'équation $(\mathscr{P}_f)$, avec les conditions initiales $\theta(0)=$ `th0` et $\dot{\theta}(0)=$ `thp0`.

Question 17

Écrire une fonction `trace_trajectoire_odeint(t_list,y_list,nom_de_fichier)` enregistrant dans `nom_de_fichier` la courbe de la trajectoire du pendule (c'est-à-dire des points $(t,\theta(t))$, où `t_list` et `y_list` sont supposées être les listes renvoyées par la fonction `odeint_f`.

Vous enverrez à l'enseignant les figures produites pour les conditions initiales $\theta(0)=1$ et $\dot{\theta}(0)=2$ et pour $n=10^2$.

Question 18

Écrire une fonction `trace_phase_odeint(t_list,y_list,nom_de_fichier)` enregistrant dans `nom_de_fichier` le portrait de phase du pendule (c'est-à-dire des points $(\theta(t),\dot{\theta}(t))$, où `t_list` et `y_list` sont supposées être les listes renvoyées par la fonction `odeint_f`.

Vous enverrez à l'enseignant les figures produites pour les conditions initiales $\theta(0)=1$ et $\dot{\theta}(0)=2$ et pour $n=10^2$.

Exercice 182 – (EQD-009)

Objectif On souhaite déterminer l'évolution d'une population d'une proie en fonction de celle de son prédateur.

On suppose un milieu où existe une population « u » de proies (lapins) interagissant avec une unique population « v » de prédateurs (renards).

Modèle sans prédation

Sans prédateur, l'évolution du nombre de proies est donné par l'équation différentielle suivante : $\frac{\mathrm{d}u(t)}{\mathrm{d}t} = a\times u(t)$ avec a le taux re reproduction des proies.

On prendra $u_0=\alpha$ et $a=\alpha 10^{-2}$.

Question 1 *Donner la population de lapins à 1 près après 20 unités de temps.*

Modèle avec prédation

Le modèle avec prédateur est donné par :

$$\begin{cases} \dfrac{du(t)}{dt} = u(t)(a - b\times v(t)) \\ \dfrac{dv(t)}{dt} = -v(t)(c - d\times u(t)) \end{cases}$$

avec :

- a : taux de reproduction des proies ;
- b : taux de mortalité des proies à cause des prédateurs ;
- c : taux de mortalité des prédateurs ;
- d : taux de reproduction des prédateurs.

On donne $u_0=\alpha$, $a=\alpha 10^{-2}$, $b=\alpha 10^{-3}$ ainsi que $v_0=\alpha+10$, $c=\alpha 10^{-2}$, $d=\dfrac{\alpha}{5}\times 10^{-3}$.

Question 2 *Donner la population de lapins après 300 unités de temps.*

Question 3 *Donner la population de renards après 300 unités de temps.*

Exercice 183 – (maitre-chien)

Un marcheur M suit une trajectoire rectiligne à vitesse constante V_M. Son chien C, qui part d'un point éloigné, court pour le rejoindre à vitesse constante V_C. À chaque instant, sa course est dirigée vers son maître, *i.e.* les vecteurs $\frac{\mathrm{d}\overrightarrow{OC}}{\mathrm{d}t}(t)$ et $\overrightarrow{CM}(t)$ sont colinéaires et de même sens : $\frac{\mathrm{d}\overrightarrow{OC}}{\mathrm{d}t}(t) = V_C\frac{\overrightarrow{CM}(t)}{\|\overrightarrow{CM}(t)\|}$. Ainsi, les coordonnées $\big(x(t),y(t)\big)$ du chien vérifient le système différentiel :

$$\begin{cases} x'(t) = V_C\dfrac{V_M\,t - x(t)}{\sqrt{\big(V_M\,t-x(t)\big)^2+\big(-y(t)\big)^2}} \\ y'(t) = V_C\dfrac{-y(t)}{\sqrt{\big(V_M\,t-x(t)\big)^2+\big(-y(t)\big)^2}} \end{cases}$$

Écrire un programme permettant de résoudre ce système différentiel avec la méthode d'Euler jusque $t=38s$ et avec 100 subdivisions.

On a $V_M = 1{,}5+\alpha/100\,\mathrm{m.s^{-1}}$ et $V_C = 8\,\mathrm{m.s^{-1}}$ et les conditions initiales : $x(0)=100+\alpha$ et $y(0)=300+\alpha$ (coordonnées de la position initiale donnée en mètre).

Question 1

Donner l'expression de la variable initiale à préciser dans la méthode d'Euler sous forme numérique en respectant son type.

Question 2

Donner $x(t=38s)$ obtenu avec la méthode d'Euler.

Question 3

Donner $y(t=38s)$ obtenu avec la méthode d'Euler.

17 Équations stationnaires

Exercice 184 – (SATIO-001)

Question 1

On rappelle que $\cos\left(\frac{\pi}{10}\right)$ est une des solutions de l'équation $16x^4-20x^2+5=0$. En déterminer une valeur approchée à 10^{-10} près.

Exercice 185 – (SATIO-002)

Question 1

Déterminer une valeur approchée du nombre d'or à 10^{-10} près.

Exercice 186 – (SATIO-003)

Question 1

Écrire une fonction déterminant, pour tout $p\in\mathbb{N}^*$, une valeur approchée de l'équation $1=x+x^2+\cdots+x^p$.

Exercice 187 – (SATIO-004)

Introduction

On s'intéresse ici au problème de régression par moindres carrés, que l'on envisagera dans le cas particulier d'un problème de cinétique chimique.

On souhaite modéliser l'évolution de la concentration d'un réactif, en fonction du temps. On a mesuré, expérimentalement, les données contenues dans la table **??**.

R En préambule de votre script, vous aurez intérêt à créer deux listes, une pour les temps, une pour les concentrations, de la manière suivante.

```
T = [0, 7, 18, 27, 37, 56, 102]
C = [34.83, 32.14, 28.47, 25.74,
    23.14, 18.54, 11.04]
```

On se donne donc une suite de sept mesures de temps $(t_i)_{i=0}^6$ ainsi que de sept mesures de concentration $(C_i)_{i=0}^6$. L'objectif est de trouver une fonction f telle que les points $f(t_i)$ sont proches des C_i. Pour quantifier cela, on considèrera le critère des moindres carrés, ce qui revient à considérer la quantité

$$\mathscr{L}(f)=\frac{1}{7}\sum_{i=0}^{6}(C_i-f(t_i))^2. \qquad \text{(MC)}$$

Il exite une infinité de fonctions qui annulent cette quantité (penser par exemple à l'interpolation de Lagrange) et qui n'ont aucun rapport avec notre problème. Nous allons donc nous restreindre à des ensembles réduits de fonctions : c'est ce que l'on appellera le *modèle*.

Modèle cinétique d'ordre 1.

Dans ce modèle, on considère que la concentration vérifie l'équation différentielle

$C'(t)=-kC(t)$ avec $k>0$, qui admet pour solution $C:t\mapsto C(0)e^{-kt}$.

Pour simplifier, on suppose que $C(0)=C_0=34{,}83$ mol.L^{-1}. On considère donc le modèle

$$\mathscr{M}_1=\left\{f_k^1:t\mapsto C_0e^{-kt}\;\middle|\;k>0\right\}.$$

On cherche donc une fonction dans $\mathscr{M}_1$ minimisant (MC), ce qui revient à chercher un paramètre $k>0$ minimisant

$$L^1(k)=\frac{1}{7}\sum_{i=0}^{6}\left(C_i-C_0\mathrm{e}^{-kt_i}\right)^2.$$

Modèle cinétique d'ordre 2.

Dans ce modèle, on considère que la concentration vérifie l'équation différentielle

$C'(t)=-kC^2(t)$ avec $k>0$, qui admet pour solution $C:t\mapsto\dfrac{C(0)}{C(0)kt+1}$

Pour simplifier, on suppose que $C(0)=C_0=34{,}83$ mol.L^{-1}. On considère donc le modèle

$$\mathscr{M}_2=\left\{f_k^2:t\mapsto\frac{C_0}{C_0kt+1}\;\middle|\;k>0\right\}.$$

On cherche donc une fonction dans $\mathscr{M}_2$ minimisant (MC), ce qui revient à chercher un paramètre $k>0$ minimisant

$$L^2(k)=\frac{1}{7}\sum_{i=0}^{6}\left(C_i-\frac{C_0}{C_0kt_i+1}\right)^2.$$

Rendu graphique.

Question 1

Écrire une fonction `trace_fonction(xmin,xmax,f,nom_de_fichi` qui trace la courbe de la fonction `f` de `xmin` à `xmax`, puis sauvegarde le résultat dans le fichier `nom_de_fichier`.

Par exemple, les commandes successives

```
xmin = 0
xmax = 20
def f(x) :
    return 0.02* x*(x-5)
trace_fonction(xmin,xmax,f,'fig_ex_fonction.png
    ')
```

devront produire (et sauvegarder) un graphique semblable (vous n'essaierez pas dans un premier temps de reproduire les titres, légendes etc.) à `fig_ex_fonction.png`, que vous trouverez sur le site de la classe.

Question 2

Écrire une fonction `trace_ajustement(X,Y,f,nom)` qui trace un nuage de points dont les abscisses sont données dans le vecteur `X` et les ordonnées dans le vecteur `Y`, qui superpose la courbe de la fonction `f` pour des arguments allant de `min(X)` à `max(X)` et qui sauvegarde l'image produite dans le fichier `nom`.

Par exemple, les commandes successives

```
import numpy as np
T = [0, 7, 18, 27, 37, 56, 102]
C = [34.83, 32.14, 28.47, 25.74, 23.14, 18.54,
    11.04]
def g(x) :
    return 34 - 0.2*x
trace_ajustement(T,C,g,'fig_ex_ajustement.png')
```

devront produire (et sauvegarder) un graphique semblable (vous n'essaierez pas dans un premier temps de reproduire les titres, légendes etc.) à `fig_ex_ajustement.png`, que vous trouverez sur le site de la classe.

Régression pour le modèle d'ordre 1.

Question 3
Écrire une fonction `L1(k)` prenant en argument un flottant positif `k` et renvoyant la valeur de $L^1(k)$.

Question 4
Représenter L^1 sur un intervalle convenablement choisi. Semble-t-il y avoir un minimum à cette fonction ? Quelle est la régularité de L^1 sur $\mathbb{R}_+^*$?
Vous enverrez la figure tracée à votre enseignant.

Question 5
Déterminer une fonction dL^1 sur laquelle appliquer une méthode de Newton permet d'obtenir le lieu du minimum de L^1. Écrire une fonction `dL1(k)` prenant en argument un flottant positif `k` et renvoyant la valeur de $dL^1(\texttt{k})$.

Question 6
Écrire une fonction `ddL1(k)` prenant en argument un flottant positif `k` et renvoyant la valeur de $(dL^1)'(\texttt{k})$.

Question 7
Écrire une fonction `newton(f, fp, x0, eps)` implémentant la méthode de Newton pour la fonction `f`, ou `fp` est supposée être la dérivée de `f`, à partir du point `x0` et s'arrêtant dès que deux points consécutifs sont distants d'au plus `eps`.

Question 8
Appliquer la méthode de Newton pour trouver le lieu du minimum de L^1, noté k_1. Converge-t-elle pour toutes les valeurs initiales ?
Dans le script, vous écrirez une fonction `val_k1()`, sans argument, effectuant ce calcul et renvoyant la valeur trouvée. On pourra affecter à la variable `k1` cette valeur pour la suite du script.

Question 9
Représenter les mesures expérimentales superposées à la fonction $f_{k_1}^1$.
Vous enverrez la figure tracée à votre enseignant.

Régression pour le modèle d'ordre 2.

Question 10
Écrire une fonction `L2(k)` prenant en argument un flottant positif `k` et renvoyant la valeur de $L^2(\texttt{k})$.

Question 11
Représenter L^2 sur un intervalle convenablement choisi. Semble-t-il y avoir un minimum à cette fonction ? Quelle est la régularité de L^2 sur $\mathbb{R}_+^*$?
Vous enverrez la figure tracée à votre enseignant.

Question 12
Déterminer une fonction dL^2 sur laquelle appliquer la méthode de la sécante (ou de Newton) permet d'obtenir le lieu du minimum de L^2. Écrire une fonction `dL2(k)` prenant en argument un flottant positif `k` et renvoyant la valeur de $dL^2(\texttt{k})$.

Question 13
Écrire une fonction `secante(f, x0, x1, eps)` implémentant la méthode de la sécante pour la fonction `f`, à partir du point `x0` et `x1` et s'arrêtant dès que deux points consécutifs sont distants d'au plus `eps`.

Question 14
Appliquer la méthode de la sécante pour trouver le lieu du minimum de L^2, noté k_2.
Dans le script, vous écrirez une fonction `val_k2()`, sans argument, effectuant ce calcul et renvoyant la valeur trouvée. On pourra affecter à la variable `k2` cette valeur pour la suite du script.

Question 15
Représenter les mesures expérimentales superposées à la fonction $f_{k_2}^2$.
Vous enverrez la figure tracée à votre enseignant.

Question 16
Facultatif : appliquer la méthode de Newton pour trouver le lieu du minimum de L^2, noté k_2. Converge-t-elle pour toutes les valeurs initiales ?
Dans le script, vous écrirez une fonction `val_k2_bis()`, sans argument, effectuant ce calcul et renvoyant la valeur trouvée.

Comparaison qualitative des deux modèles.

Question 17
Commenter les résultats des deux parties précédentes, ainsi que l'utilisation des méthodes de Newton et de la sécante.

Facultatif : régression linéaire.

Dans le modèle d'ordre 1, on peut remarquer que l'on a $-\log\left(\frac{C(t)}{C(0)}\right) = kt$. On a donc ici une relation linéaire entre le temps et une transformation des concentrations. On peut donc effectuer une *régression linéaire par moindres carrés*, qui consiste à minimiser le critère

$$L^1_{\text{lin}}(k) = \frac{1}{7}\sum_{i=0}^{6}\left(kt_i + \log\left(\frac{C_i}{C(0)}\right)\right)^2.$$

On effectue en réalité une régression linéaire par moindres carrés sur les points de coordonnées

$$\left(t_i, \log\left(\frac{C_i}{C(0)}\right)\right).$$

Question 18
Montrer qu'il existe un unique paramètre k_1' minimisant L^1_{lin}, puis le déterminer explicitement (en fonction des données du problème).

Question 19
Écrirer une fonction `val_kp1()`, sans argument, et renvoyant la valeur de k_1'. On pourra affecter ensuite à la variable `kp1` cette valeur pour la suite du script.

Question 20
Représenter les mesures expérimentales superposées à la fonction $f_{k_1'}^1$.
Vous enverrez la figure tracée à votre enseignant.

Question 21
Comparer cette méthode avec celle utilisée dans la partie 187. Quels sont les avantages de chacune ?

Exercice 188 – (SATIO-005)

Question 1
Donner une approximation de $\int_\alpha^{\alpha+1}\cos(\sqrt{t})\,dt$ avec la mé-

thode des trapèzes et 1000 subdivisions.

Question 2
Donnez une approximation (à 10^{-5} près) de l'unique réel positif solution de l'équation $x^2+\sqrt{x}-10=\alpha$ avec la méthode de votre choix.

Question 3
Donnez le nombre d'itérations nécessaires pour obtenir ce résultats avec la méthode de Newton en prenant pour valeur initiale α.

Question 4
Donnez le nombre d'itérations nécessaires pour obtenir ce résultats avec la méthode de Dichotomie sur l'intervalle $[0,12+\alpha]$.

Question 5
Donnez à l'aide une approximation (à 10^{-5} près) de l'unique réel positif t tel que $\int_{\alpha}^{\alpha+t}(2+\sqrt{x}+\cos x)\,dx=10$. *Pour l'intégration numérique, on pourra utiliser la méthode des trapèzes avec 1000 subdivisions.*

Question 6
Donner une valeur approchée de $x(1)$ avec x l'unique fonction vérifiant $x(0)=\alpha$ et pour tout $t\in\mathbb{R}$, $x'(t)=3\cos(x(t))+t$.

Question 7
Donner une valeur approchée de $x\left(1+\frac{\alpha}{10}\right)$ avec x l'unique fonction vérifiant $x(0)=0$, $x'(0)=0$ et pour tout $t\in\mathbb{R}$, $x''(t)=1+\sin(t+x(t))$.

Question 8
Donner une approximation de $\beta\in\mathbb{R}$, pour que l'unique solution de l'équation différentielle non linéaire $x''(t)=1+\arctan(t+x(t))$ avec les conditions initiales $x(0)=0$ et $x'(0)=\beta$ vérifie $x\left(1+\frac{\alpha}{10}\right)=1+\frac{2}{3}\alpha$.

18 Systèmes d'équations

Exercice 189 – (polynome)

On souhaite déterminer une parabole passant les points d'équation $y=ax^2+bx+c$ de la parabole passant par les points $(-1,9)$, $(1,3)$ et $(2,3)$.

Question 1
Poser le système d'équations à résoudre.

Question 2
Déterminer les coefficients a, b et c par l'utilisation du pivot de Gauss.

Question 3
Sur un même graphe faire apparaître les 3 points ainsi que la paraboles obtenue par interpolation.

Exercice 190 – (SYS-000)

Soit A une matrice carrée telle que toute sous-matrice principale (sous-matrice carrée calée dans le coin supérieur gauche) soit inversible.

On rappelle que l'opération $L_i \leftarrow L_i+\alpha L_j$ effectuée sur les lignes d'une matrice A, revient à effectuer le produit $T(i,j,\alpha)\times A$, où $T(i,j,\alpha)$ est la matrice de transvection égale à l'identité, à l'exception du coefficient (i,j) qui vaut α.

Et l'opération $C_i \leftarrow C_i+\alpha C_j$ effectuée sur les colonnes d'une matrice A, revient à effectuer le produit $A\times T(j,i,\alpha)$. On peut montrer qu'alors la matrice A admet une décomposition $A=LU$, où L est une matrice triangulaire inférieure et U une matrice triangulaire supérieure. La matrice L est obtenue en appliquant à la matrice identité les opérations sur les colonnes correspondant aux inverses des matrices d'opérations sur les lignes permettant de trianguler A par la méthode de Gauss (sans échange de ligne : toute sous-matrice de A étant principale, on peut montrer que si la matrice A est rendue triangulaire pour ses k premières colonnes, alors le coefficient diagonal de la colonne suivante n'est pas nul et peut être choisi comme pivot).
Plus précisément : si T_1, ... , T_p sont les p transvections successives à appliquer à A pour la rendre triangulaire supérieure, nous obtenons : $(T_p\times\ldots\times T_1)\times A=U$. Il faut alors poser $L=(T_p\times\ldots\times T_1)^{-1}=I_n\times T_1^{-1}\times\ldots\times T_p^{-1}$.

L'intérêt réside dans le fait qu'on ramène la résolution de $AX=B$ à la résolution de 2 systèmes triangulaires. Ceci s'avère notamment intéressant lorsqu'on a plusieurs systèmes à résoudre associés à la même matrice A.

Nous allons écrire un programme permettant de calculer cette décomposition LU, et le comparer à l'algorithme du pivot de Gauss classique.
Pour cela, nous écrirons les matrices sous forme de tableaux bi-dimensionnels de type `array`, en utilisant le module `numpy`, dont voici quelques exemples d'utilisations :

```
import numpy as np
A=np.array([[1, 2], [3, 4]])
B=np.eye(2)                          # B = I_2
B[1,0]=2
C = A.dot(A) + 2*B.dot(A)            # calcule C = A^2 + 2BA.
```

Nous utiliserons aussi les fonctions définies dans le fichier `random_matrix`, disponible sur le site de la classe, ainsi que la fonction `clock` du module `time` pour chronométrer les temps d'éxécution.

1. Écrire une fonction `trans_ligne` qui à un quadruplet `(A,i,j,alpha)`, où A est une matrice d'ordre n, $i,j\in[\![1,n]\!]$, et $\alpha\in\mathbb{R}$, associe le résultat du produit de la matrice de transvection $T(i,j,\alpha)$ d'ordre n, par A. Attention : par souci d'efficacité, il est préférable de voir cette opération comme une opération entre des lignes de A, plutôt que comme un produit matriciel. On s'arrangera autant que possible pour éviter les calculs inutiles sur les coefficients qu'on sait être nuls.
2. Écrire une fonction `trans_colonne` qui à un quadruplet `(A,i,j,alpha)`, où A est une matrice d'ordre n, $i,j\in[\![1,n]\!]$, et $\alpha\in\mathbb{R}$, associe le résultat du produit de A par l'inverse de la matrice de transvection $T(i,j,\alpha)$ d'ordre n, par A (la remarque de la question précédente est toujours d'actualité).

3. Écrire une fonction `LU` retournant les deux matrices L et U de la décomposition ci-dessus, pour une matrice A donnée.
4. Écrire une fonction `resolution_sup` de résolution d'un système triangulaire supérieur.
5. Écrire une fonction `resolution_inf` de résolution d'un système triangulaire inférieur.
6. Soient $A \in \mathcal{M}_{100}(\mathbb{R})$ et $b_0, \dots, b_{24} \in \mathcal{M}_{100,1}(\mathbb{R})$, choisies aléatoirement (les coefficients étant pris dans $[\![0,100]\!]$).
   Calculer :
   (a) le temps de résolution de 25 systèmes associés à A, en calculant une fois pour toutes la décomposition LU de A, puis en résolvant 2 systèmes triangulaires pour chaque second membre B ;
   (b) le temps de résolution de 25 systèmes associés à A, en reprenant un pivot complet pour chaque système ;
   (c) le temps de résolution de 25 systèmes associés à A, en calculant A^{-1} par la méthode du pivot, une fois pour toutes puis en calculant $A^{-1}B$ pour chaque second membre.
7. Plus généralement, tracer, pour $k \in [\![1,25]\!]$, les 2 courbes correspondant au temps de réponse pour la résolution de k systèmes associés à $A \in \mathcal{M}_{100}(\mathbb{R})$, par décomposition LU, et par calcul de A^{-1}. Commenter.

Exercice 191 – (SYS-001)

On considère une fonction réelle f de classe $\mathcal{C}^n$ au voisinage de 0, vérifiant $f(0)=0$ et $f'(0)\neq 0$. On cherche à calculer le développement limité à l'ordre n de f^{-1} au voisinage de 0, en fonction de celui de f.

On représentera un polynôme par la liste de ses coefficients, écrits par degrés croissants. Ainsi, le polynôme $1+2X^2$ pourra être représenté par les listes `[1.,0.,2.]` et `[1.,0.,2.,0.,0.]`.

Les vecteurs et matrices seront représentés en `Python` en utilisant le type `array` de la bibliothèque `numpy`.

Chaque fois que l'on demandera de calculer une complexité, on rédigera ceci *sur papier*, après avoir recopié *à la main* le code de la fonction et numéroté ses lignes, afin de pouvoir y faire référence clairement. Les fonctions ayant une complexité asymptotique clairement sous-optimale seront fortement pénalisées. On ne demande pas de justifier ce dernier point.

Enfin, on s'interdira les spécificités de Python permettant d'éviter d'écrire des boucles. Notamment : la fonction `sum`, la méthode `.dot`, l'écriture d'une liste en compréhension. Chaque boucle sera accompagnée d'un invariant justifiant sa correction.

Question 1
Écrire une fonction `produit(P,Q,n)` prenant en argument deux listes de flottants `P` et `Q` ainsi qu'un entier n et renvoyant la liste des coefficients du produit des polynômes représentés par `P` et `Q`, tronquée à l'ordre n.

Par exemple, avec `P = [1.,1.,1.]` et `Q = [3.,-1.,2.,4.]`, le produit des polynômes représentés par `P` et `Q` est

$$3+2X+4X^2+5X^3+6X^4+4X^5.$$

Ainsi, `produit(P,Q,3)` renverra `[3.0, 2.0, 4.0, 5.0]`, tandis que `produit(P,Q,7)` renverra `[3.0, 2.0, 4.0, 5.0, 6.0, 4.0, 0., 0.]`.

Question 2
Question manuscrite. Étudier la complexité asymptotique de `produit(P,Q,n)` en fonction de n.

Question 3
Question manuscrite. On note le développement limité de f^{-1} au voisinage de 0 et à l'ordre n comme suit :

$$f^{-1}(t) \underset{t\to 0}{=} x_1 t + x_2 t^2 + \cdots + x_n t^n + o(t^n).$$

En écrivant, au voisinage de 0, $(f^{-1}\circ f)(t) = t$, écrire le système vérifié par le vecteur $(x_1,\dots,x_n)$ et justifier que ce système est triangulaire inférieur. On pourra introduire les coefficients des développements limités des f^j, pour $1 \leq j \leq n$, au voisinage de 0 et à l'ordre n.

Question 4
Écrire une fonction `matrice(P)` renvoyant la matrice du système précédent, où `P` est la liste des coefficients de la partie principale du développement limité de f au voisinage de 0.

Par exemple, si $f(t) \underset{t\to 0}{=} 3t-5t^2+3t^4+o(t^4)$, alors on utilisera `P = [0.,3.,0.,-5.,3.]` et l'on remarquera que `P` est alors de longueur 5.

Question 5
Question manuscrite. Étudier la complexité asymptotique de `matrice(P)` en fonction de n, où $n+1$ est la longueur de `P`.

Question 6
Écrire une fonction `resoutTI(T,Y)` renvoyant la solution du système $TX=Y$, où `T` est une matrice triangulaire inférieure et `Y` un vecteur de même dimension que `T` n'a de lignes.

Question 7
Question manuscrite. Étudier la complexité asymptotique de `resoutTI(T,Y)` en fonction de n, où n est la dimension de `Y`.

Question 8
Écrire une fonction `DLreciproque(P)` renvoyant la liste des coefficients du développement limité en 0 de f^{-1}, de même ordre que celui de f, où `P` est la liste des coefficients de la partie principale du développement limité de f au voisinage de 0.

Question 9
Question manuscrite. Étudier la complexité asymptotique de `DLreciproque(P)` en fonction de n, où $n+1$ est la longueur de `P`.

Question 10
Application. Écrire une fonction `DLtan(n)` prenant en argument en entier naturel `n` et renvoyant la liste des coefficients de la partie principale de développement limité de la fonction tangente au voisinage de 0 et à l'ordre n.

Commenter le résultat obtenus pour $n=7$.

Exercice 192 – (SYS-002)

On étudie dans ce problème le comportement d'un objet déformable (poutre 1D) fixé à une extrémité et soumis à l'action d'une force à son autre extrémité.

On modélise cela par le montage en série de n systèmes ressort-ammortisseur (voir figure **??**) présentant, pour tout $0 \leq i < n$, les mêmes masses m, raideurs k et coefficients d'amortissement c. On note $f(t)$ l'intensité de la force appliquée au dernier élément du système.

Pour tout $0 \leq i < n$ et tout temps t, on note $u_i(t)$ le déplacement de l'élément n° i par rapport à sa position d'équilibre et l'on introduit le vecteur

$$X(t) = \begin{pmatrix} u_0(t) \\ \vdots \\ u_{n-1}(t) \end{pmatrix}.$$

On peut montrer que X vérifie l'équation différentielle

$$MX'' + CX' + KX = F(t),$$

où

$$M = \begin{pmatrix} m & 0 & \cdots & \cdots & 0 \\ 0 & m & 0 & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & m & 0 \\ 0 & 0 & \cdots & 0 & m \end{pmatrix} \quad , \quad C = \begin{pmatrix} 2c & -c & 0 & \cdots & 0 \\ -c & 2c & -c & \ddots & \vdots \\ 0 & -c & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2c & -c \\ 0 & \cdots & 0 & -c & 2c \end{pmatrix}$$

$$K = \begin{pmatrix} 2k & -k & 0 & \cdots & 0 \\ -k & 2k & -k & \ddots & \vdots \\ 0 & -k & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2k & -k \\ 0 & \cdots & 0 & -k & 2k \end{pmatrix} \quad , \quad F(t) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ f(t) \end{pmatrix}$$

Les matrices carrées écrites ici sont de dimension $n \times n$ et le vecteur $F(t)$ est de dimension n.

On utilise la méthode d'Euler pour résoudre de manière approchée cette équation différentielle. On fixe donc un pas de temps `dt` > 0 et l'on calcule de proche en proche les

$$X_q = X(q \times \texttt{dt}).$$

Le système est au repos au départ, on suppose donc que $X_0 = X_{-1} = 0$.

Avec

$$H = \frac{1}{\texttt{dt}^2} M + \frac{1}{\texttt{dt}} C + K$$

et, pour tout $q \geq 1$,

$$G_q = \frac{1}{\texttt{dt}^2} M(2X_{q-1} - X_{q-2}) + \frac{1}{\texttt{dt}} CX_{q-1} + F(q \times \texttt{dt}),$$

on montre que, pour tout $q \geq 1$,

$$H \times X_q = G_q.$$

On remarquera que, comme les matrices C et K, H est tridiagonale.

Pour mettre en œuvre la méthode d'Euler, on résout donc plusieurs systèmes de même membre de gauche H. Nous allons utiliser ici la méthode de Jacobi pour résoudre ce système.

Question 1

Écrire une fonction `matrice_H(m,c,k,dt,n)` prenant en argument quatre flottants strictement positifs `m`, `c`, `k` et `dt` ainsi qu'un entier strictement positif `n` et renvoyant la matrice H de dimension `n` $\times$ `n` définie ci-dessus.

On donne la fonction suivante, permettant de calculer le vecteur G_q.

```
import numpy as np

def vecteur_G(m,c,dt,q,X1,X2,f):
    """Renvoie le vecteur Gq de dimension n
    Préconditions : m,c,dt flottants strictements posi
                    q entier positif
                    X1,X2 vecteurs nx1 (X1 : X_q-1 et
                    f fonction réelle, appel en O(1)"""
    n,_ = X1.shape
    G = np.zeros((n,1))
    for i in range(n):
        G[i] = m*(2X1[i]-X2[i])/(dt**2)
    G[0] = G[0] +  (2c * X1[0] - c*X1[1])/dt
    G[-1] = G[-1] +  (2c * X1[-1] - c*X1[-2])/dt + f(q
    for i in range(1,n-1) :
        G[i] = G[i] + (2c * X1[i] - c*X1[i-1] - c*X1[i
    return G
```

Question 2

Étudier la complexité temporelle d'un appel de la fonction `vecteur_G(m,c,dt,q,X1,X2,f)`, en fonction d'une grandeur que l'on explicitera.

La méthode du pivot de Gauss étant numériquement peu stable, on lui préfère souvent des méthodes itératives. On expose ici celle de Jacobi.

On résout le système $n \times n$ écrit matriciellement : $H \times Y = G$, à partir d'un vecteur initial Y^0. Pour tout $q \in \mathbb{N}$, on note

$$Y^q = \begin{pmatrix} y_0^q \\ \vdots \\ y_{n-1}^q \end{pmatrix}.$$

Si le vecteur Y^q est construit, la méthode de Jacobi consiste à définir le vecteur Y^{q+1} par, pour tout $0 \leq i < n$,

$$y_i^{q+1} = \frac{1}{h_{i,i}} \left(g_i - \sum_{j \neq i} h_{i,j} y_j^q \right).$$

Question 3

Proposer une simplification de la somme écrite ci-dessus utilisant le fait que la matrice H utilisée ici est tridiagonale (on distinguera les cas $i = 0$ et $i = n-1$).

Question 4

Écrire une fonction `iteration_Jacobi(H,G,Yq)` prenant en argument une matrice H tridiagonale ainsi que deux vecteurs `G` et `Yq`, renvoie le vecteur Y^{q+1} décrit ci-dessus.

On s'interdira ici d'utiliser la multiplication matricielle fournie par la bibliothèque `numpy`.

On considère la norme euclidienne d'un vecteur :

$$\left\| \begin{pmatrix} x_0 \\ \vdots \\ x_{n-1} \end{pmatrix} \right\| = \sqrt{x_0^2 + \cdots + x_{n-1}^2} = \sqrt{\sum_{k=0}^{n-1} x_k^2}.$$

Question 5
Écrire une fonction `carre_norme(X)` prenant en argument un vecteur `X` et renvoyant le carré de sa norme, *i.e.* $\|X\|^2$.

On utilise le critère d'arrêt suivant pour la méthode de Jacobi : on fixe une précision $\varepsilon > 0$, on s'arrête dès que $\|HY_q - G\| \leq \varepsilon \|G\|$ et l'on prend Y_q comme approximation de la solution de ce système.

Question 6
Écrire une fonction `Jacobi(H,G,Y0,eps)` prenant en argument une matrice tridiagonale `H`, deux vecteurs `G` et `Y0` ainsi qu'un flottant strictement positif `eps` et renvoyant la valeur approchée du système $H \times X = G$ avec la condition initiale `Y0` et le critère d'arrêt donné par $\varepsilon =$ `eps`.

La méthode de Jacobi converge si la matrice H est à diagonale strictement dominante, *i.e.* si pour tout $0 \leq i < n$,

$$|h_{i,i}| > \sum_{j \neq i} |h_{i,j}|.$$

Question 7
Justifier que la méthode de Jacobi est ici convergente.

N'attaquez la dernière question que si vous avez traité toutes les autres questions correctement.

Question 8
On effectue ce calcul sur p valeurs de temps distinctes. On rappelle que l'on résout donc p systèmes de la forme $H \times X_q = G_q$. Toujours en utilisant la méthode de Jacobi, peut-il être préférable de mettre en œuvre une autre stratégie de résolution, du point de vue de la complexité temporelle ?

On pourra supposer que tous les appels de la fonction `Jacobi` *ont la même complexité temporelle.*

Exercice 193 – (SYS-003)

Question 1
Résoudre le système

$$\begin{pmatrix} 5 & 8 & -2 \\ 3 & 1 & 5 \\ 0 & -2 & 6 \end{pmatrix} \times X = \begin{pmatrix} 21 \\ 16 \\ 10 \end{pmatrix}.$$

On pose

$$A = \begin{pmatrix} 3 & -2 & 5 \\ -4 & 1 & 1 \\ 2 & 3 & -2 \end{pmatrix}.$$

Question 2
En effectuant un seul calcul, résoudre simultanément les systèmes :

$$AX = \begin{pmatrix} 20 \\ -2 \\ -7 \end{pmatrix}, \quad AX = \begin{pmatrix} -21 \\ 23 \\ -1 \end{pmatrix}, \quad AX = \begin{pmatrix} -12 \\ 17 \\ 4 \end{pmatrix}, \quad AX = \begin{pmatrix} 6 \\ -2 \\ 3 \end{pmatrix}.$$

Exercice 194 – (SYS-004)

Question 1
Inverser la matrice

$$A = \begin{pmatrix} 5 & -3 & 2 & 1 & -1 \\ 3 & 6 & 8 & 1 & -3 \\ 5 & 6 & 3 & 0 & 2 \\ 4 & 6 & 2 & 8 & 3 \\ -6 & 3 & 5 & -1 & -2 \end{pmatrix}.$$

Exercice 195 – (SYS-005)

On considère les points de coordonnées

$$M_1 \begin{pmatrix} -5 \\ 11,67 \end{pmatrix}, \quad M_2 \begin{pmatrix} -2 \\ 4,52 \end{pmatrix}, \quad M_3 \begin{pmatrix} 1 \\ -0,15 \end{pmatrix}, \quad M_4 \begin{pmatrix} 2 \\ -3,31 \end{pmatrix}.$$

On cherche à ajuster une droite affine sur le nuage de points (M_1, M_2, M_3, M_4) par le critère des moindres carrés. Avec

$$A = \begin{pmatrix} 1 & -5 \\ 1 & -2 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}, \quad X = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 11,67 \\ 4,52 \\ -0,15 \\ -3,31 \end{pmatrix},$$

Question 1
déterminer X minimisant la quantité

$$\|AX - B\|.$$

Question 2
Produire une figure superposant les points (M_1, M_2, M_3, M_4) et la droite d'équation $y = \alpha + \beta x$.

Exercice 196 – (SYS-006)

On considère les points de coordonnées

$$M_1 \begin{pmatrix} -2 \\ 7,62 \end{pmatrix}, \quad M_2 \begin{pmatrix} -1 \\ 3,87 \end{pmatrix}, \quad M_3 \begin{pmatrix} 0 \\ 0,94 \end{pmatrix}, \quad M_4 \begin{pmatrix} 1 \\ 1,56 \end{pmatrix}, \quad M_5 \begin{pmatrix} 2 \\ 2,66 \end{pmatrix}.$$

On cherche à ajuster une courbe polynomiale de degré 2 sur le nuage de points $(M_1, M_2, M_3, M_4, M_5)$ par le critère des moindres carrés. Avec

$$A = \begin{pmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix}, \quad X = \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 7,62 \\ 3,87 \\ 0.94 \\ 1,56 \\ 2,66 \end{pmatrix},$$

Question 1
déterminer X minimisant la quantité

$$\|AX - B\|.$$

Question 2
Produire une figure superposant les points $(M_1, M_2, M_3, M_4, M_5)$ et la courbe d'équation $y = \alpha + \beta x + \gamma x^2$.

Exercice 197 – (SYS-007)

Dans cette partie, on travaille avec la matrice

$$A = \begin{pmatrix} 0 & 1 & 32 & 243 \\ 1 & 32 & 243 & 1024 \\ 32 & 243 & 1024 & 3125 \\ 243 & 1024 & 3125 & 7776 \end{pmatrix}$$

de terme général $a_{ij} = (i+j-2)^5$ pour $1 \leq i, j \leq n$ (attention en Python, les indices commencent à 0 et le terme général est alors $(i+j)^5$).

Question 1

Résoudre

$$A \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ \alpha \end{pmatrix}$$

Donner la valeur de x_1.

Question 2

Calculer $B = A^3$ et donner le reste du coefficient de B situé sur la première ligne et la première colonne de B (donc d'indices 0 et 0 en `numpy`) dans la division par $10\,000 + \alpha$.

19 Bases de données

Exercice 198 – (SQL-000)

Les classes de MPSI du lycée LMM organisent un tournoi d'intelligences artificielles de morpion. Une partie de morpion se déroule sur un damier de trois cases par trois cases. Un premier joueur choisit une case. Puis, le deuxième joueur choisit une autre case. Puis, le premier joueur choisit une troisième case, et l'on continue ainsi jusqu'à ce que :

- soit un joueur aligne trois cases, dans ce cas il gagne ;
- soit le damier est rempli, dans ce cas il y a match nul.

Les professeurs de ces classes ont donc créé une base de données afin de gérer ce tournoi. Cette base contient deux tables :

- une table contenant les informations des joueurs : identifiant du joueur, pseudonyme, date de naissance, classe ;
- une table contenant les informations des parties : identifiant de la partie, identifiant du joueur n° 1, identifiant du joueur n° 2, résultat de la partie, coups joués, date de la partie.

Voici les instructions SQL ayant permis de créé ces deux tables.

```
CREATE TABLE joueurs (
         idj INTEGER,
         pseudo VARCHAR(50),
         datenaiss DATE,
         classe VARCHAR(50),
         PRIMARY KEY (idj)
         );

CREATE TABLE parties (
         idp INTEGER,
         idj1 INTEGER,
         idj2 INTEGER,
         res INTEGER,
         coups INTEGER,
         date DATETIME,
         PRIMARY KEY(idp),
         FOREIGN KEY(idj1) REFERENCES joueurs,
         FOREIGN KEY(idj2) REFERENCES joueurs
         );
```

Quelques conventions et rappels.

- Si le résultat d'une partie vaut 0, il y a match nul, s'il vaut 1 le joueur 1 a gagné et s'il vaut 2 le joueur 2 a gagné.
- Pour une partie, on lit dans l'entier `coups` les coups effectués par les joueurs. Ainsi, si `coups = 54892`, alors
  - le joueur 1 joue dans la case 5 ;
  - le joueur 2 joue dans la case 4 ;
  - le joueur 1 joue dans la case 8 ;
  - le joueur 2 joue dans la case 9 ;
  - le joueur 1 joue dans la case 2.

  Ainsi, dans cette partie, le joueur 1 a gagné (alignement sur la deuxième colonne).
- Les dates (type `DATE`) sont au format `AAAA-MM-JJ`.
- Les dates-heures (type `DATETIME`) sont au format `AAAA-MM-JJ HH:MM:SS`.
- Les dates et dates-heures sont écrites comme des chaînes de caractères (exemple : `"1515-09-13"`) et peuvent être comparées entre elles.
- Si une requête ne donne pas qu'une réponse (exemple : plusieurs lignes vérifient un critère), on inscrira toutes les réponses.
- Pour effectuer une jointure d'une table `T` avec elle même, on pourra lui donner un alias avec l'instruction `AS`. Voici un exemple.

```
SELECT *
FROM T
JOIN T AS Tbis ON T.[...] = Tbis.[...] ;
```

Question 1
Quel est le pseudonyme de l'étudiant d'identifiant n° 42 ?

Question 2
Quel est l'identifiant de l'étudiant dont le pseudonyme est `"Galois"` ?

Question 3
Combien de parties ont été jouées ?

Question 4
Combien y a-t-il eu de matchs nuls ?

Question 5
Combien de parties différentes (*i.e.* de successions de coups différentes) ont été jouées ?

Question 6
Quel est l'identifiant de l'étudiant le plus jeune ? Les donner tous s'il y en a plusieurs.

Question 7
Combien de parties ont été jouées par l'étudiant de pseudonyme `"Tux"` ?

Question 8
Combien de parties ont été perdues par l'étudiant de pseudonyme `"Tux"` ?

Question 9
Combien de parties ont été gagnées par des étudiants de MPS1 contre des étudiants de MPS2 ?

Question 10
Quels sont les pseudonymes des joueurs ayant joué le plus de parties en tant que joueur n° 1 ? Les donner tous s'il y en a plusieurs.

Question 11
Combien de parties ont été jouées en sept coups ou moins, en 2018 et entre deux étudiants d'une même classe ?

Exercice 199 – (SQL-001)

La base de données[7] `medocs.sqlite` contient 5 tables :

LABORATOIRES : Contient une liste de laboratoires pharmaceutiques. Cette table possède deux attributs :
- **id :** Identifiant du laboratoire dans la base de données.
- **laboratoire :** Nom du laboratoire.

7. Source : http ://base-donnees-publique.medicaments.gouv.fr/telechargement.php

CIS_COMPO : Contient la liste des compositions qualitatives et quantitatives des médicaments de la base de données, substance par substance. Cette table contient 8 attributs :
- **code_CIS :** Code CIS (code identifiant de spécialité) d'un médicament, vous pouvez le considérer comme un identifiant de médicament.
- **désignation :** Désignation de l'élément pharmaceutique.
- **code_substance :** Code de la substance.
- **dénomination_substance :** Dénomination de la substance.
- **dosage :** Dosage de la substance.
- **ref_dosage :** Référence de ce dosage. Exemple : `"(pour) un comprimé"`.
- **nature_compo :** Nature du composant (principe actif : « SA » ou fraction thérapeutique : « ST »).
- **numéro_liaison :** Numéro permettant de lier, le cas échéant, substances actives et fractions thérapeutiques.

HAS_Liens : Contient les liens vers les avis de la commission de transparence (CT) de la Haute Autorité de la Santé (HAS). Cette table contient 2 attributs :
- **code_HAS :** Code de dossier HAS.
- **lien :** Lien vers la page d'avis de la CT.

CIS_bdpm : Cette table contient la liste des médicaments commercialisés, ou en arrêt de commercialisation depuis moins de trois ans. Cette table contient 11 attibuts :
- **code_CIS :** Code CIS (code identifiant de spécialité) d'un médicament, vous pouvez le considérer comme un identifiant de médicament.
- **dénomination :** Dénomination du médicament.
- **forme :** Forme pharmaceutique.
- **voie :** Voies d'administration (avec un séparateur «; » entre chaque valeur quand il y en a plusieurs).
- **statut :** Statut administratif de l'autorisation de mise sur le marché (AMM).
- **procédure :** Type de procédure d'autorisation de mise sur le marché (AMM).
- **commercialisation :** État de commercialisation.
- **date_AMM :** Date d'AMM (format JJ/MM/AAAA).
- **statutBdM :** Valeurs possibles : « Alerte » (icône rouge) ou « Warning disponibilité » (icône grise).
- **numéro_autorisation :** Numéro de l'autorisation européenne.
- **titulaire :** Numéro du laboratoire titulaire.
- **surveillance :** Surveillance renforcée (triangle noir) : valeurs « Oui » ou « Non ».

CIS_HAS_SMR : Cette table contient l'ensemble des avis de SMR (Service médical rendu) de la HAS. Cette table contient 6 attributs :
- **code_CIS :** Code CIS (code identifiant de spécialité) d'un médicament, vous pouvez le considérer comme un identifiant de médicament.
- **code_HAS :** Code de dossier HAS.
- **motif :** Motif d'évaluation.
- **date_avis :** Date de l'avis de la Commission de la transparence (format AAAAMMJJ).
- **valeur :** Valeur du SMR.
- **libellé :** Libellé du SMR.

Question.

Question 1
`code_CIS` est-il une clé primaire de la table `CIS_bdmp` ?

Question 2
`code_HAS` est-il une clé primaire de la table `CIS_HAS_SMR` ?

Question 3
Donner le nom du laboratoire dont le numéro d'identification est α.

Question 4
Donner le nombre de médicaments produits par ce laboratoire.

Question 5
Donner le nombre médicaments de ce laboratoire dont l'AMM a été donnée le 1er janvier 2000 ou après.

Question 6
Donner le code CIS du médicament de ce laboratoire ayant la plus ancienne AMM. S'il y en a plusieurs, on donnera le code CIS le plus petit.

Question 7
Quel est le lien internet vers la page d'avis de la CT sur ce dernier médicament (on ne donnera que la série de chiffres à la fin de cette adresse, qui sont au nombre de 6 ou 7 suivant les adresses) ?

Question 8
Quelle est la somme des numéros de liaison des médicaments de ce laboratoire ?

Question 9
Donner le code CIS du médicament de ce laboratoire ayant le plus de substances différentes. S'il y en a plusieurs, on donnera le code CIS le plus petit.

Question 10
Cette question peut se traiter en interrogeant la base de données depuis `Python` : combien de codes CIS contiennent votre numéro α comme sous-chaîne ? Si votre α est compris entre 0 et 9, vous le ferez précéder d'un 0. Par exemple les α valant 8 ou 27 sont contenus dans 60008927 mais pas dans 60002875.

Exercice 200 – (SQL-002)

Un professeur d'informatique a créé une base de données pour gérer les notes de ses interrogations hebdomadaires. Pour cela, il a créé trois tables : `etudiants`, `interros`, `notes`. Voici les commandes SQL ayant permis de créer ces tables.

```
CREATE TABLE etudiants (
    -- table des données des étudiants
    id INTEGER, -- identifiant de l'étudiant
␣␣␣␣nom␣VARCHAR␣NOT␣NULL,␣--␣nom␣de␣l'étudiant
    prenom VARCHAR NOT NULL, -- prénom de l'é
        tudiant
␣␣␣␣date_naissance␣DATE␣NOT␣NULL,␣--␣date␣de␣
    naissance␣de␣l'étudiant, format AAAA-MM-JJ
```

```
    PRIMARY KEY (id)
    );

CREATE TABLE interros (
    -- table des données des interros
    id INTEGER, -- identifiant de l'interro
␣␣␣␣titre␣VARCHAR,␣--␣titre␣de␣l'interro
    sujet VARCHAR, -- sujet de l'interro
␣␣␣␣date␣DATE,␣--␣date␣du␣jour␣où␣a␣été␣donnée␣
    l'interro, format AAAA-MM-JJ
    PRIMARY KEY (id)
    );

CREATE TABLE notes (
    -- table des notes des étudiants aux
        interros
    id_etudiant INTEGER NOT NULL, -- identifiant
         de l'étudiant
␣␣␣␣id_interro␣INTEGER␣NOT␣NULL,␣--␣identifiant
    ␣de␣l'interro
    note INTEGER NOT NULL, -- note obtenue
    PRIMARY KEY (id_etudiant, id_interro),
    FOREIGN KEY (id_etudiant) REFERENCES
        etudiants,
    FOREIGN KEY (id_interro) REFERENCES interros
    );
```

Question 1
Peut-on donner plusieurs notes au même étudiant et pour la même interrogation ?

Question 2
Donner une requête SQL traduisant l'opération suivante, exprimée dans le vocabulaire d'algèbre relationnelle usuel :

$$\pi_{\text{sujet}}(\sigma_{\text{id}=1}(\text{interros})).$$

Question 3
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants de la classe.

Question 4
Écrire une requête SQL permettant d'obtenir la liste des prénoms des étudiants de la classe, sans doublon.

Question 5
Les enfants du professeur se sont amusés à rentrer des données factices, que le professeur aimerait retrouver afin de les effacer ensuite. Écrire une requête SQL permettant d'obtenir la liste des identifiants des étudiants ayant pour nom `"reinedesneiges"`.

Question 6
Écrire une requête SQL permettant d'obtenir la date de naissance de l'étudiant le plus jeune de la classe.

Question 7
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants ayant obtenu au moins un 20 à une des interrogations.

Question 8
Écrire une requête SQL permettant d'obtenir la liste des noms, prénoms d'étudiants et titres d'interrogations pour chaque note de 0 obtenue.

Question 9
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants, avec la moyenne des notes de chaque étudiant.

Question 10
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants, suivis pour chaque étudiant du nombre d'interrogations rendues.

Question 11
Écrire une requête SQL permettant d'obtenir le nom, le prénom et le nombre de copies rendues par l'étudiant ayant rendu le plus de copies (on suppose qu'il n'y en a qu'un). On rappelle que l'instruction `LIMIT k` permet de tronquer une table à ses `k` premières lignes.

Question 12
Écrire une requête SQL permettant d'obtenir la liste des noms et prénoms des étudiants ayant rendu au moins 10 copies, suivis du nombre de copies rendues.

Question 13
Écrire une requête SQL permettant d'obtenir la liste des titres des interrogations, suivie pour chaque interrogation du nombre d'étudiants qui n'ont pas rendu de copies.

Question 14
Écrire une requête SQL permettant d'obtenir la liste des titres des interrogations pour lesquelles tous les étudiants ont rendu une copie.

Exercice 201 – (SQL-003)

On s'intéresse dans cet exercice à la gestion des données produites par la caisse enregistreuse d'une boulangerie (fictive), un jour donné. Sur chaque ticket produit par la caisse figure une ligne par produit vendu, indiquant notamment le nombre d'unités vendues par produit et le prix de chaque produit.

Chaque étudiant dispose d'un fichier `bdd_boulangerie_`α`.sqlite` possédant trois tables. Voici les commandes ayant permis de créer ces tables.

```
CREATE TABLE tickets (
        -- table des tickets
        id INTEGER,
        heure TIME NOT NULL,
        paiement VARCHAR(10) NOT NULL,
        PRIMARY KEY (id)
        );

CREATE TABLE produits (
        -- table des produits
        id INTEGER,
        nom VARCHAR(50) NOT NULL,
        prix FLOAT,
        PRIMARY KEY (id)
        );

CREATE TABLE lignes_tickets (
        -- table des lignes des tickets
        id INTEGER,
        idt INTEGER,
        idp INTEGER,
        quantite INTEGER NOT NULL,
        PRIMARY KEY (id),
```

```
FOREIGN KEY (idt) REFERENCES tickets,
FOREIGN KEY (idp) REFERENCES produits
);
```

La table `produits` recence les produits vendus par la boulangerie et indique pour chaque produit son nom et son prix (en €).

La table `tickets` recence pour chaque ticket l'heure d'enregistrement du ticket et le moyen de paiment (carte bleue, liquide ou chèque).

La table `lignes_tickets` indique pour chaque ticket (identifiant `idt`) et chaque produit (identifiant `idp`) le nombre d'unités du produit acheté sur ce ticket.

Question 1
Combien d'unités ont été vendues par la boulangerie ce jour là ?

Question 2
Quel est le chiffre d'affaire de la boulangerie ce jour là ?

Question 3
Quel est l'identifiant du produit dont ont été vendues le plus d'unités ? S'il y en a plusieurs, mettez le plus petit.

Question 4
Combien de tickets ne contiennent qu'un produit vendu (quelqu'en soit le nombre d'unités) ?

Question 5
Combien de tickets ne contiennent que des produits vendus en une unité ?

Question 6
Quel est l'heure du dernier ticket enregistré ?

Question 7
Quelle est la valeur en € du premier ticket enregistré ?

La banque de la boulangerie prélève une commission de 1% sur chaque transaction effectuée par carte bleue.

Question 8
Quel est le montant total prélevé par la banque à la boulangerie ce jour là ?

Question 9
Combien y a-t-il de tickets dont la valeur est supérieure ou égale à 10 € ?

Exercice 202 – (SQL-004)

Donner des requêtes pour :

- trouver le nombre d'acteurs qui ont joué au moins deux rôles ;
- trouver les acteurs qui sont aussi des réalisateurs ;
- trouver les acteurs nés le 1er janvier 1930 ou après ;
- compter les acteurs nés avant le 1er janvier 1940 ;
- trouver les acteurs qui ont joué au moins dans un film où jouait Martin Freeman ;
- trouver les réalisateurs qui ont dirigé Clint Eastwood.

Exercice 203 – (SQL-005)

Base de données des Pokemon

Nous allons utiliser la base de données issue du site `http://veekun.com/`. Un fichier nommé « veekun-pokedex.sqlite » doit être présent sur le bureau de votre ordinateur. Ouvrir cette base de données avec DB Browser for SQLite.

Structure de la table de données

Question 1
En utilisant DB Browser for SQLite, donner le nombre de tables contenu dans la base de données.

Dans un premier temps, nous allons utiliser uniquement la table `pokemon` qui répertorie les pokémons.

Question 2
Donner le schéma relationnel de cette table. On le donnera sous la forme nom_table(attribut_1 : type, attribut_2 : type, ...).

Question 3
Donner la définition d'une clé primaire.

Table des pokemons

Question 4
Quelle est la taille de pikachu ?

Question 5
Quelle est le poids de pikachu ?

Question 6
En utilisant une des valeurs précédentes, quels pokemons sont plus grands (strictement) que pikachu ?

Question 7
Sans utiliser une des valeurs précédentes, quels pokemons sont plus grands (strictement) que pikachu ?

Question 8
Combien y a t-il de pokemons plus grands (strictement) que pikachu ?

Question 9
Combien de pokemons ont la même taille que pikachu (lui y compris) ?

Question 10
Parmi les pokemons ayant la même taille que pikachu, donner le nom et le poids du plus gros.

Question 11
Donner le nom et la taille et le poids du plus grand pokemon.

Question 12
Quel pokémon est le plus petit ? (Il peut y en avoir plusieurs ...)

Question 13
Lister le nombre de pokemons par taille en les classant du plus grand au plus petit.

Question 14
Quel est le nombre maximal de pokemons ayant la même taille ? Donner la taille et le nombre.

Question 15
Quels est le nom et la taille du second pokemon le plus grand ?

Question 16
Quelle est la taille moyenne des pokemons ? (deux décimales après la virgule).

Notre niveau d'expérience permet d'attraper des pokemons de hauteur égale à 8 à 0.5 près (inclus). Nous souhai-

terions donc savoir combien de pokemons pourront être attrapés sans changer ces réglages.

Question 17
Combien de pokémons sont capturés avec le réglage par défaut ?

Classement des pokemons

Maintenant nous souhaitons placer les pokemons sur une carte selon leurs propriétés, pour cela nous allons utiliser les tables suivantes.

La table `pokemon_species` contient les colonnes :

- `id` (clé primaire) : identifiant du pokemon ;
- `identifier` : nom du pokemon ;
- `generation_id` : identifiant de génération qui correspond aussi au numéro de la génération ;
- `habitat_id` : identifiant d'habitat ;

D'autres attributs existent, mais ils ne seront pas utilisés dans notre étude.

La table `pokemon_habitats` contient les colonnes :

- `id` (clé primaire) : identifiant d'habitat ;
- `identifier` : nom de l'habitat ;

Question 18
Écrire la requête SQL permettant d'afficher le nom du pokemon et le nom de son habitat.

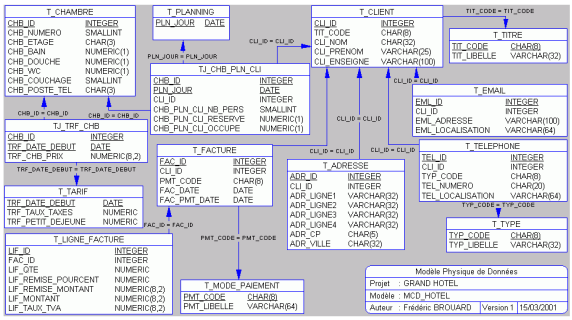
Question 19
Combien de pokemons vivent en forêt ('forest' en anglais) ?

Question 20
Combien de pokemons de la generation 3 vivent en forêt ('forest' en anglais) ?

Exercice 204 – (hotel)

On donne le schéma d'une base de données d'un hôtel.

Le modèle physique de la base de données d'un hotel est le suivant.



La plupart des clefs sont des entiers (I) qui pourront être auto générés par exemple par auto-incrément. Pour certaines entités, notamment celles servant de références à la saisie (`MODE_PAIEMENT`, `TYPE`) la clef est un code. Enfin pour les entités `TARIF` et `PLANNING`, il a été choisi une date comme clef. Chaque entité est repérée à l'aide d'un trigramme (code de 3 lettres) qui sert de préfixe pour chaque attribut. Exemple : `CHB` pour `CHAMBRE`, `LIF` pour `LIGNE_FACTURE`, etc... Les booléens seront représentés par des valeurs numériques 0 (faux) et 1 (vrai), chaque attribut ayant obligatoirement une valeur par défaut. L'association « occupée » permet de connaître la réservation ou l'occupation d'une chambre (une chambre peut avoir été réservée mais pas occupée), c'est pourquoi cette association possède les attributs `NB_PERS` (nombre de personnes : entier) `RESERVE` (réservée : booléen) et `OCCUPE` (occupe : booléen).

Une chambre à une date donnée, ne peut être occupée que par un seul client. Mais un client peut occuper plusieurs chambres à la même date ou la même chambre à différentes dates, voire même plusieurs chambres à plusieurs dates...

Définition des entités :

- l'entité `CLIENT` : un client peut avoir plusieurs adresses, plusieurs numéros de téléphone et plusieurs e-mail. Pour le téléphone, comme pour l'e-mail, l'attribut `localisation` permet de savoir si le téléphone est situé au domicile, à l'entreprise, etc...
- l'entité `TITRE` permet de donner un titre à une personne, parmi les valeurs `M.` (monsieur), `Mme.` (madame) et `Melle.` (mademoiselle) ;
- l'entité `TYPE` permet de connaître le type de téléphone, parmi les valeurs `TEL` (téléphone), `FAX` (télécopie) et `GSM` (portable) ;
- l'entité `MODE_PAIEMENT` permet de connaître le genre de paiement, parmi les valeurs `ESP` (espèces), `CHQ` (chèque), `CB` (carte bancaire). L'association « payée » intègre la date du paiement d'une facture.

Travail demandé

Question 1 *Donner la requête permettant de lister tous les noms, prénoms et les titres (M. Mme. ou Melle.) des clients.*

Question 2 *Donner le nombre de clients enregistrés dans la base.*

Question 3 *Trouver les noms et prénoms des clients dont le titre est 'Mme.' (madame).*

Question 4 *Donner les noms , prénoms et les titres (Mme. ou Melle.) des clientes.*

Question 5 *Donner le nombre de clientes.*

Question 6 *Classer par ordre alphabétique les clients de sexe féminin. On ne demande que les noms et les prénoms qui devront être appelés Noms et Prénoms.*

Question 7 *Établir une nouvelle relation (table) faisant apparaître les noms des clients et leurs numéros de téléphone.*

Question 8 *Donner la liste des clients ayant le même nom*

Question 9 *Donner le nombre d'occurrences de chacun de ces noms.*

Question 10 *Donner la valeur moyenne des remises en pourcentage et en montant.*

Question 11 *Donner la valeur maximale des remises*

en pourcentage et en montant.

Question 12 *Donner les identifiants facture (*`FAC_ID` *que l'on renomera* `fac_id1`*) qui ont bénéficié de remise (soit en pourcentage soit en montant).*

Question 13 *Donner les identifiants clients (*`CLI_ID`*) qui ont bénéficié d'une remise (soit en pourcentage soit en montant).*

Question 14 *Donner les identifiants clients (*`CLI_ID`*) qui n'ont pas bénéficié d'une remise.*

Question 15 *Donner le nom et prénom du client qui a le plus bénéficié de remises. Donner le montant de cette remise totale.*

20 Problèmes

20.1 Introduction : profil de puissance record

La puissance mécanique est une grandeur qui quantifie une variation instantané de travail mécanique.

Elle est très souvent utilisée par les cycliste pour estimer ses capacités ou planifier des entrainement. Un indicateur de performance intéressant à mettre en oeuvre est le profil de puissance record.

Il est alors intéressant pour définir le profile d'un coureur de tracer ce graphe. Pour cela il faut analyser des zone d'efforts similaire puis estimer une puissance moyenne sur la durée correspondante. On obtient alors un point sur le graphe.

Analyse préliminaire : filtrage numérique

20.2 Filtre numérique passe bas du premier ordre

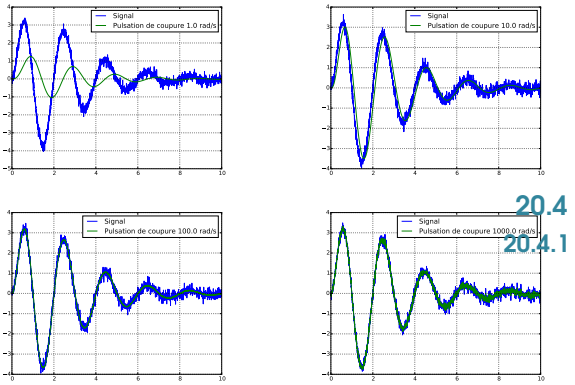
Soit un filtre linéaire du premier ordre. Son équation différentielle est de la forme :

$$s(t)+\tau\frac{\mathrm{d}s(t)}{\mathrm{d}t}=Ke(t)$$

En utilisant un schéma d'Euler implicite, on a l'approximation suivante : $\frac{\mathrm{d}s(t)}{\mathrm{d}t}=\frac{s_k-s_{k-1}}{h}$. En conséquences,

$$s_k+\tau\frac{s_k-s_{k-1}}{h}=Ke_k \Leftrightarrow s_k=\frac{hKe_k+\tau s_{k-1}}{h+\tau}$$

La fréquence d'échantillonnage est ici de 1 KHz. Le pas de dérivation est de 0,001 s. On réalise alors différents filtrages en faisant varier la pulsation de coupure du filtre.



On observe que lorsque la pulsation de coupure diminue, le signal est de plus en plus lissé, limitant ainsi le bruit. En revanche, le signal est atténue et déphasé.

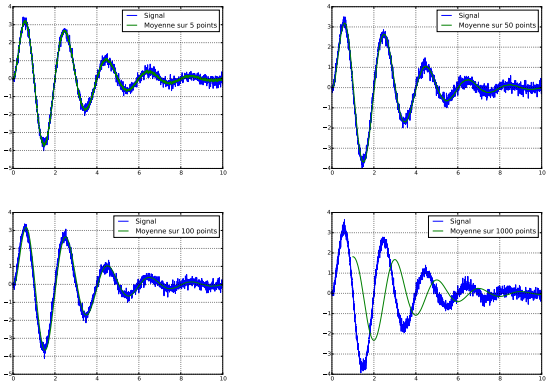
On donne l'algorithme permettant de réaliser ce filtre à partir d'un tableau "$signal$" contenant des valeurs pour chaque instant t correspondant au tableau "t" avec une fréquence de coupure "$freq$".

```
def filtrage_passe_bas(freq,t,signal):
    tau = 1/freq # Coupure du filtre
    K=1
    res=[signal[0]]
    for i in range(1,len(signal)):
        h=t[i]-t[i-1]
        res.append((h*K*signal[i]+tau*res[-1])/(
            h+tau))
    return res
```

20.3 Filtrage numérique par moyenne glissante

Une autre solution pour lisser une courbe est de réaliser une moyenne glissante. Si on réalise ce filtrage en temps réel, cela signifie qu'un point lissé à l'échantillon n est la moyenne des $n-1$ échantillons précédents et de l'élément en cours.

Il faut donc attendre l'acquisition de $n-1$ échantillons avant de disposer de la courbe.



On donne l'algorithme permettant de réaliser ce filtre à partir d'un tableau "$signal$" contenant des valeurs pour chaque instant t correspondant au tableau "t" avec une taille de fenêtre de filtrage "$freq$".

```
def filtrage_moyenne(signal,fenetre):
    filtrageg =signal[0:fenetre-1]
    for i in range(fenetre-1,len(signal)):
        s = sum(signal[i-fenetre+1:i+1])/fenetre
        filtrageg=np.concatenate([filtrageg,np.
            array([s])])
    return filtrageg
```

Analyse des performances d'un cyclisme

20.4 Analyse d'un fichier gpx

20.4.1 Lecture du fichier

Question 16

Ecrire un programme permettant de savoir si un mot se trouve dans une chaine de caractère.

Un fichier gpx permet d'écrire l'ensemble des données issues d'un appareil du type gps ou montre connecté. Il recueille les 3 coordonnées spatiales (altitude par rapport au niveau de la mer, latitude et longitude) en fonction du temps. Ce fichier contient donc un ensemble de coordonnées spatiales en fonction du temps. Ouvrir le fichier gpx et analyser son contenu. On remarquera en particulier que le fichier est organisé d'un certaine manière. Ce fichier comporte différentes balises [8].

Le fichier (< gpx >) peut contenir :

- Des métadonnées (< $metadata$ >), décrivant le contenu du fichier GPX par entre autre :
  - un nom (< $name$ >)
  - une description (< $desc$ >)

8. source : wikipedia : `https://fr.wikipedia.org/wiki/GPX_(format_de_fichier)`

Modéliser

  - l'auteur du fichier (< $author$ >) comprenant son nom, une adresse mail et un lien vers son site web.
  - ...
- Une liste de traces ou track (< trk >) chacune décrite par :
  - un nom (< $name$ >);
  - le type d'itinéraire(< $type$ >)
  - des segments de trace (< $trkseg$ >), le passage d'un segment à un autre indique une extinction du récepteur GPS ou une perte de réception. Un segment de trace est constitué :
    * d'une liste ordonnée de points de trace (< $trkpt$ >) dans laquelle figure respectivement la latitude ('lat') ainsi que la longitude ('long') en °.
    * avec son altitude par rapport au niveau de la mer en mètres (< ele >)
    * un horodatage (< $time$ >)

Question 17 20.5.1

Écrire un programme permettant de stocker sous la forme de 4 tableaux nommés tp, lat, $long$ et lat représentant respectivement pour chaque point de mesure le temps en s par rapport à l'instant initial, la latitude en °, la longitude en ° ainsi que l'altitude par rapport au niveau de la mer en m.

20.5 Superposition du tracé sur une carte

Pour des traces petites (comme un circuit) on peu assimiler les longitude et latitude à des coordonnées cartésiennes et il n'est donc pas nécessaire de réaliser des projections cartographiques.

Question 18

A l'aide des fonctions contenues dans le module "$matplotlib.pyplot$" tracer un graphe donnant la liste des points des différentes coordonnées extraites précédemment en mettant en abscisses les valeurs de longitude et en ordonnées celles de latitude. On obtient alors la trace gps du parcours réalisé.

Pour superposer cette trace à une carte, on peut utiliser le module "$folium$" qu'il faut installer avec la commande :

```
pip install folium
```

Puis importer avec :

```
import folium
```

La fonction "Map" permet de créer un objet carte centrée avec en point possédant une latitude et une longitude. Dans les arguments de cette fonction, il faut préciser les coordonnées en latitude et en longitude (en °) ainsi le niveau d'agrandissement initial avec l'entier z.

```
macarte = folium.Map(location=[lat,long],
    zoom_start=z)
```

Etant donnée une trace GPS comme celle importée précédemment, on peut connaître les latitudes min,max et les longitudes min,max qui lui correspondent. On peut se servir de ces dernière pour définir la carte initiale.

Pour superposer à cette carte la trace gps extraite précédemment on peut utiliser la fonction "$PolyLine$" qui est utilisable de la façon suivante :

```
folium.PolyLine(points).add_to(macarte)
```

Où la variable "$points$" est une liste de tuple contenant pour chaque coordonnée du fichier gpx respectivement la latitude et la longitude.

La méthode "save" appliqué à l'objet "macarte" et prenant en argument une chaine de caractère correspondante au nom du fichier "html" permet de créer une page html représentant la carte définie précédemment.

Question 19

mettre en oeuvre les fonctions du module "$folium$" pour créer une page HTML superposant la trace GPS sur une carte avec un centrage et un niveau d'agrandissement satisfaisant. Vous pourrez visualiser la page créée en l'ouvrant un navigateur internet (Firefox, Google Chrome, etc...)

Traitement du fichier

On donne l'expression de la vitesse et de l'accélération en coordonnées sphérique d'un point matériel M dans son mouvement par rapport au référentiel terrestre R_0 :

$$\overrightarrow{V}(M/R_0) = (\dot{r})\,\overrightarrow{e}_r + \left(r\cdot\dot{\theta}\right)\overrightarrow{e}_\theta + \left(r\cdot\sin\theta\,\dot{\varphi}\right)\overrightarrow{e}_\varphi$$

$$\overrightarrow{a}(M/R_0) = \left(\ddot{r} - r\cdot\dot{\theta}^2 - r\cdot\sin^2\theta\,\dot{\varphi}^2\right)\overrightarrow{e}_r + \left(2*\dot{r}\dot{\theta} + r\cdot\ddot{\theta} - r\cdot\sin\theta\right.$$

On peut relier les paramètres r, θ et φ aux coordonnées gps :

$$\begin{cases} r = alt + R \\ \theta = \dfrac{\pi}{2} - lat \\ \varphi = long \end{cases}$$

avec R le rayon de la terre supposé constant et égal à $6371km$

Question 20

Écrire un programme prenant en argument un vecteur $y(t)$ dépendant du temps ainsi qu'un vecteur t de mêmes longueurs et renvoyant le vecteur dy correspondant à l'estimation de $\dfrac{dy(t)}{dt}$ par différence finie. On pourra la noter $dif(y,t)$.

Question 21

Appliquer cette méthode pour donner des estimation de $\dfrac{dr}{dt}$, ... que pouvons-nous dire quand à la qualité de l'estimation de la dérivée.

Question 22

Écrire une fonction $vitesse$ prenant en argument les tableaux r, $theta$ et phi et renvoyant 4 tableau Vr, $Vtheta$

et $Vphi$ et V correspondant respectivement aux 3 coordonnées du vecteurs vitesse dans le système de coordonnées sphérique ainsi que la norme du vecteur vitesse.

Question 23

Mettre en évidence à l'aide d'un tracé l'éventuelle corrélation entre la vitesse du cycliste et le profil en altitude du circuit. 20.6.5

20.6 Modélisation des performances mécaniques

20.6.1 Modélisation de la puissance

En considérant le cyclique comme un solide à masse ponctuelle (cela revient à négliger les effets d'inertie de la rotation des roues) on peut estimer la puissance que le cycliste doit développer en mouvement à l'aide du théorème de l'énergie cinétique :

$$P_c = P_{inertie} + P_{aero} + P_{Pes} + P_{roul}$$

- $P_{inertie}$ est la puissance que le cycliste doit développer pour vaincre les effets d'inertie et elle provient de la variation de l'énergie cinétique.
- P_{aero} est la puissance qui résulte des effets aéronautique.
- P_{pes} est la puissance qui provient de l'action mécanique de pesanteur.
- P_{roul} est la puissance due à la résistance au roulement.

On se placera dans le référentiel terrestre supposé galiléen (R_0). Le cycliste ayant réalisé l'activité possède une masse $M = 70kg$. On prendra l'accélération de la pesanteur $g = 9,81m \cdot s^{-2}$.

20.6.2 Puissance due à la résistance au roulement

Question 24

Déterminer une fonction Proul qui prend en argument la masse d'un cycliste, un vecteur V comprenant la norme de la vitesse de déplacement du cycliste $(V = \| \overrightarrow{V}(M/R_0) \|)$ associé à une trace GPX, la pression dans les pneus et qui renvoie la puissance instantanée de roulement.

20.6.3 Puissance due aux effets d'inertie

20.7

20.7.1

La puissance due aux effets d'inerte peut s'estimer de la façon suivante :

$$P_{inertie} = M \cdot V \cdot \frac{dV}{dt}$$

Question 25

Déterminer une fonction $Pinertie$ prenant en arguments la masse d'un cycliste, un vecteur V comprenant la norme de la vitesse de déplacement du cycliste $(\|V(M/R_0)\|)$ associé à une trace GPX, la pression dans les pneus et qui renvoie la puissance instantanée de roulement.

20.6.4 Puissance aéronautique

Question 26

Déterminer une fonction $Paero$ prenant en arguments, la vitesse du vent (Vv),sa direction (α_v), le produit $C_d \times A_p$, un vecteur V comprenant la norme de la vitesse de déplacement du cycliste $(\|V(M/R_0)\|)$ associé à une trace GPX et renvoyant l'estimation de la puissance aérodynamique instantanée.

Puissance due aux effets de pesanteur

La puissance de pesanteur va dépendre du dénivelé qui correspond à la variation de la pente en fonction de la distance parcouru.

Question 27

Écrire une fonction permettant de calculer l'intégrale par rapport au temps d'une fonction $f(t)$. Elle prendre en arguments deux tableaux de même dimension f et t et renverra l'estimation de $I = \int_{t'=0}^{t} f(t')dt'$ sous la forme d'un tableau ayant la même dimension que t et f.

Question 28

Utiliser cette fonction pour donner une estimation de la distance parcouru en fonction du temps que l'on notera x.

La pente instantanée peut s'estimer par $\frac{dr(t)}{dx}$.

Question 29

Écrire une fonction prenant en argument les tableaux r et x et renvoyant l'estimation de la pente sous la forme d'un tableau de même dimension. Pour vérifier la cohérence de cette estimation tracer sur une même figure deux graphes représentant en fonction du temps $r(t)$ et $\frac{dr(t)}{dx}$.

La puissance de pesanteur peut s'estimer de la façon suivante :

$$P_{pes} = M \cdot g \cdot \sin \arctan\left(\frac{dr(t)}{dx}\right) \cdot \| \overrightarrow{V}(M/R_0) \|$$

Question 30

Déterminer une fonction $Ppes$ prenant en arguments, la masse du cycliste, la coordonnées $r(t)$, un vecteur V comprenant la norme de la vitesse de déplacement du cycliste $(\| \overrightarrow{V}(M/R_0) \|)$ associé à une trace GPX et renvoyant l'estimation de la puissance de pesanteur instantanée.

Puissance totale et profil de puissance record

Puissance totale

Question 31

A l'aide des question précédente tracer en fonction du temps les différentes puissances instantanées. Estimer la puissance totale instantanée que doit fournir et la superposer sur ce graphe.

Question 32

Comment interpréter les valeurs négatives. On veillera à ne pas en tenir compte pour la suite.

L'énergie dépensée sur le circuit par le cycliste peut s'estimer de la façon suivante :

$$E = \int_{t=0}^{t_f} P_c(t')dt'$$

Question 33

Mettre en oeuvre la méthode des trapèze pour calculer l'énergie totale consommée par le cycliste.

Modéliser

Question 34
A partir de l'estimation de la puissance instantanée précédente déterminer plusieurs point sur le PPR.

20.8 Références

- Cours de Xavier Pessoles support de cours de PSI* La Martinière Monplaisir.
- Patrick Beynet, *Supports de cours de TSI 2*, Lycée Rouvière, Toulon.
- David Crochet (créer à partir de KmPlot), CC-BY-SA-3.0, via Wikimedia Commons `https://upload.wikimedia.org/wikipedia/commons/6/6f/Fourier_d%27un_carr%C3%A9.svg`.