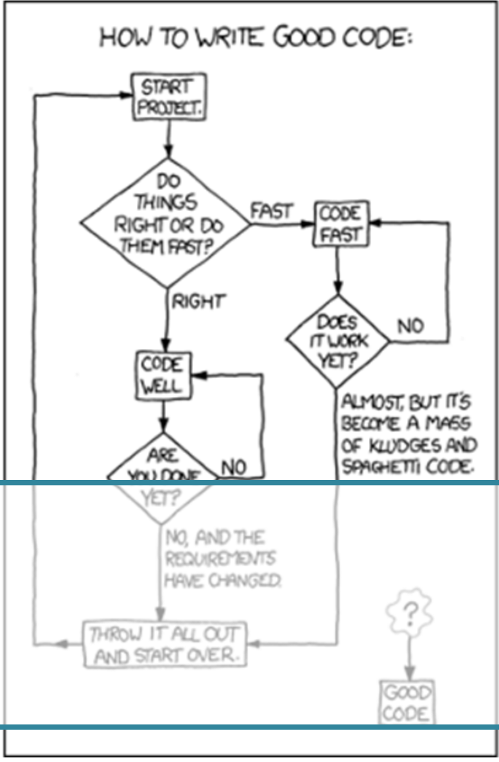

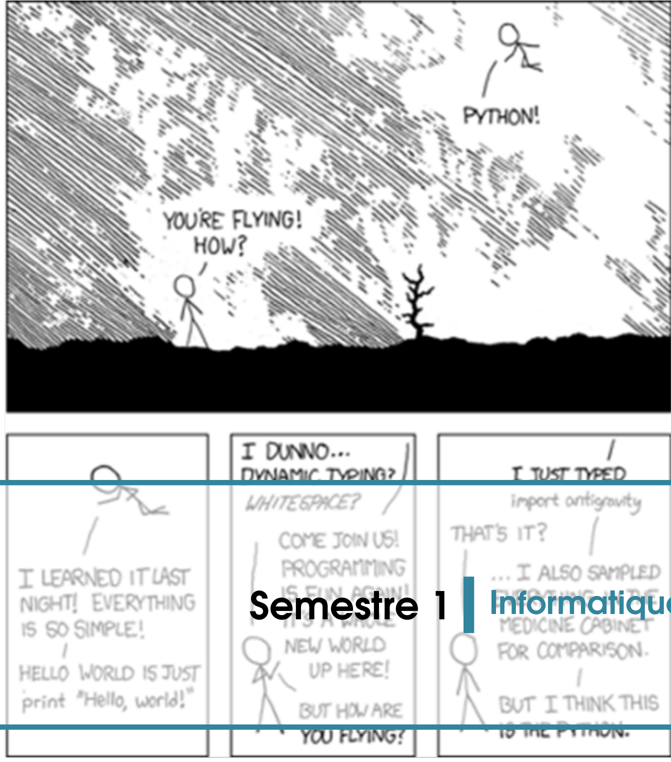

MPSI

Semestre 1 | Informatique

Thèmes d'étude

J.-P. Berne – E. Durif – X. Pessoles
MPSI

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^{54}
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
```

Modéliser

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.

======= 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`

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) $2\sqrt{2}$

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` $\leftarrow$ `a` *pour dire que l'on affecte la valeur* `a` *à la variable* `var`.

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

Question 5 *On part des affectations suivantes :* `a` $\leftarrow$ `5` *et* `b` $\leftarrow$ `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
```

```
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.

Modéliser

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* `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")
```

Modéliser

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, $0,25\times 10^{-2}$), 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.*

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 $\mathtt{p} = 4\times 10^5$, *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.*

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

$$\sum_{1\le i,j\le n} \frac{1}{i+j^2}, \qquad (1)$$

$$\sum_{1\le i<j\le n} \frac{1}{i+j^2}. \qquad (2)$$

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 $|u_n - 1| \le$ `eps`.

Exercice 60 –

Question 1
Écrire une fonction `comb(n,p)` calculant $\binom{n}{p}$ pour deux entiers naturels n, p vérifiant $0 \le p \le n$, 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 $H_n \ge$ `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).

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 `p` $= 4 \times 10^5$, *le taux est* `t` $= 0,25 \times 10^{-2}$ *et la mensualité est* `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.

Modéliser

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 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 :

$$\text{entrée : t=}\begin{matrix}[\,[1, & 0, & 0, & 1, & 0, & 0, & 1],\\ [0, & 0, & 0, & 0, & 0, & 0, & 0],\\ [1, & 0, & 0, & 0, & 0, & 0, & 0],\\ [0, & 0, & 0, & 0, & 0, & 0, & 0],\\ [0, & 1, & 0, & 0, & 0, & 0, & 1],\\ [1, & 0, & 0, & 0, & 1, & 0, & 1]\,]\end{matrix}$$

sortie : 4.

EXEMPLE 2 :

$$\text{entrée : t=}\begin{matrix}[\,[0, & 0, & 0, & 1, & 1, & 1, & 1],\\ [0, & 0, & 0, & 1, & 1, & 1, & 1],\\ [0, & 0, & 0, & 1, & 1, & 1, & 1],\\ [1, & 1, & 1, & 1, & 1, & 0, & 0],\\ [1, & 1, & 1, & 1, & 1, & 0, & 0]\,]\end{matrix}$$

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

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.

Modéliser

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

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

On dit qu'un nombre x s'insère en position $i \in [0, n-1]$ dans le tableau `t` si $t_i \leq x \leq t_{i+1}$. 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 $n \geq 2$ un entier. Un diviseur strict de n est un entier $1 \leq d \leq n-1$ 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 : $\mathtt{k}=\underline{a_{\mathtt{n}-1}\dots a_1 a_0}_2$, c'est-à-dire que

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

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):
```

Modéliser

```
    """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 2$^{\text{nde}}$ 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 :
$H(p) = \frac{a_m p^m + \dots + a_1 p + a_0}{b_n p^n + \dots + b_1 p + b_0}$ 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 $G_{\max} - 3$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 ($G_{\mathrm{dB}}(0) = 0$) non résonant. La fonction G_{dB} est monotone décroissante et l'équation $G_{\mathrm{dB}}(\omega) + 3 = 0$ 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.
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
```

- `.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
```

1.2 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.*

1.3 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* `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.

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

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 bidimensionnels) 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.

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_fichier)` *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.

Modéliser

$v_1, \cdots, u_{n-1}+v_{n-1})$, 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`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 $t = [t_0, \dots, t_{n-1}]$ d'entiers de longueur n, on appelle sous-tableau croissant de t un tableau $[t_{i_0}, \dots, t_{i_{k-1}}]$ (donc, de longueur k), avec

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

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 $(e_0,\dots,e_{n-1}) \in \{0,1\}^{[\![0,n[\![}$, avec, si $0 \le i < n$, $e_i = 1$ 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.

```python
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 : $\mathtt{k} = \underline{a_{\mathtt{n}-1} \dots a_1 a_0}_2$, c'est-à-dire que

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

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 $0 \le i < n$, 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 $0 \le i < n-1$, 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 $n \le 511$.

Exercice 105 –

Le but de cet exercice est de construire la suite ordonnée des entiers de la forme $2^p 3^q 5^r$, 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*	*13*	*14*	*15*	*16*
				↑ 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 $2^p 3^q 5^r$.
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**	**0**
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**
indice	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*

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 $1 \le j < k < i < n$ (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 $t = [t_0, \ldots, t_{n-1}]$, 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 $\mathtt{p}^2$, 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 $\mathtt{t} = [t_0, \ldots, t_{n-1}]$ 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

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

et si

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

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

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

`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).
- 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 ?*

1.6 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, 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.

Modéliser

```python
    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

```python
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

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

Question 4
(Question facultative) Écrire une fonction `sauve_echiquier(p, 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_entree, 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.

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.*

Modéliser

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.

$$
\begin{array}{c}
\underline{\mathbf{2}} \\
4\ \underline{\mathbf{5}} \\
1\ 7\ \underline{\mathbf{8}} \\
2\ 3\ 1\ \underline{\mathbf{6}} \\
9\ 4\ 6\ \underline{\mathbf{5}}\ 2
\end{array}
$$

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 :

2	2	2	2	26
4 5	4 5	4 5	20 24	
1 7 8	1 7 8	12 16 19		
2 3 1 6	11 9 7 11			
9 4 6 5 2				

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 :

```python
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=[t_0,t_1,...,t_{n-1}]`, et on appelle *tranche* de `t` toute sous-tableau non vide `[t_i,t_{i+1},...,t_{j-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 élements 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

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

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».

```python
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 :

```python
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

Modéliser

(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 défini 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`

```python
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.*

```python
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.*

```python
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`

```python
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
```

Modéliser