TD 2

Types simples

Savoirs et compétences :

- ❒ *AA.C4 : Comprendre un algorithme et expliquer ce qu'il fait*
- ❒ *AA.C5 : Modifier un algorithme existant pour obtenir un résultat différent*
- ❒ *AA.C6 : Concevoir un algorithme répondant à un problème précisément posé*
- ❒ *AA.C7 : Expliquer le fonctionnement d'un algorithme*
- ❒ *AA.C9 : Choisir un type de données en fonction d'un problème à résoudre*
- ❒ *AA.S6 : Variables : notion de type et de valeur d'une variable, types simples.*
- ❒ *AA.S7 : Expressions et instructions simples*
- ❒ *AA.S10 : Notion de fonction informatique*

Évaluer les expressions suivantes en repérant auparavant celles qui donnent des résultats de type <class '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`

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) $(2^7+2^4+2) \div 2^{10}$

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) $2^{(10^4)}$

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`

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
```

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

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)`

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`

Dans IDLE, 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 (touche F5). Observez le résultat dans l'interpréteur interactif. Modifiez les instructions pour que tous les résultats de calcul s'affichent dans l'interpréteur interactif. QuestionDans 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.

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

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",""))`

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
```

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")`

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
```

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])`

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` ?

QuestionQuel 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.

Dans les cas où c'est possible, affecter les valeurs aux variables correspondantes à l'aide de l'interpréteur interactif d'IDLE. 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"`

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 d'IDLE.

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`

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)
```

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`.

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
```

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 ?

Mêmes questions en remplaçant suffisantes par nécessaires. Supposons que la variable x est déja 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 ? 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
```

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

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

QuestionÉcrire une fonction qui prend en argument une liste `moy_extr(L)` et renvoie en sortie la moyenne du premier et du dernier élément de `L`. Écrire une fonction `L` 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*. Écrire une fonction `norme` qui prend en argument un entier `lettre` et renvoie la `i`^{e} lettre de l'alphabet. Écrire une fonction `i` qui prend en argument un entier naturel `carres` et qui renvoie la liste des `n` premiers carrés d'entiers, en commençant par 0. QuestionOn 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 `n`, qui ne modifie pas la liste initiale et renvoie en sortie une nouvelle liste.

b) Écrire une telle fonction `incr_sans_effet_de_bord`, qui modifie la liste initiale et ne renvoie rien en sortie (ponctuer par un `incr_avec_effet_de_bord`).

Question Écrire une fonction `return None` prenant en argument

- un couple de nombres `appartient(a,c,r)`;
- un couple de nombres `a`;
- un nombre positif `c`;

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

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.

QuestionIndenter de deux manières différentes la suite d'instructions suivante afin que la variable `r` contienne `t` pour une indentation, puis `True` pour l'autre.

```
False
```

QuestionRéécrire la suite d'instructions suivante de manière plus appropriée.

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

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

a) Écrire une fonction qui renvoie la négation du booléen `neg(b)` sans utiliser `b`.

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

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

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.

```
and
```

QuestionQue fait la fonction suivante ? La corriger pour qu'elle coïncide avec le but annoncé.

```
from random import randrange
L = [randrange(100) for i in range(100)] # 100 valeu
m = 50
for x in L:
if x <= m:
p = x
print(p)
```

Décrire ce que fait cette suite d'instructions.

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

Et celle-ci ?

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

Écrire une suite d'instructions permettant de calculer la somme des racines carrées des cinquante premiers entiers naturels non nuls. Écrire la suite d'instructions suivantes dans un fichier, l'enregistrer puis l'exécuter (F5). À l'instant qui vous convient, presser Ctrl+C.

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

QuestionUn 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

```
a = 1
while a>0:
    a = a+1
```
ne prenant pas d'argument et renvoyant le résultat demandé.

QuestionÉcrire une fonction `nb_exos()` prenant en argument un entier naturel et renvoyant sa racine carrée comme un entier si c'est un carré parfait, comme un flottant sinon. QuestionDans 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 `racine(n)` et `n`. Un joueur tire sur la case `a`.

Écrire une fonction `b` qui renvoie un booléen indiquant si le navire est touché ou non. QuestionUn 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 `x` 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 `touche(a,b,x)` renvoyant la somme totale (mensualités plus le dernier paiement) que vous aurez payé au banquier.

c) Écrire une fonction `reste_a_payer(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.

QuestionUn 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 `somme_totale_payee(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?

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

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.* QuestionÉcrire une fonction `cout_total(p, t, m, d)` renvoyant $\binom{n}{p}$ (nombre de combinaisons de p éléments parmi n). On pourra bien entendu introduire une fonction auxiliaire[1]. QuestionOn appelle suite de Fibonacci la suite F définie par $F_0=0$, $F_1=1$ et pour tout $n\in\mathbb{N}$, $F_{n+2}=F_{n+1}+F_n$.

Écrire une fonction `duree_mensualite(p,t,m)` 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). QuestionOn pose $u_0=1$ et pour tout $n\in\mathbb{N}$,

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

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

Écrire une fonction `comb(p,n)` renvoyant la valeur de v_n.

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

Vous pouvez calculer u_n pour de grandes valeurs de n.

Attention : on fera attention à ce que le calcul de `fib(n)` de demande pas trop de (re)calculs inutiles. Pour fixer les idées, vous pouvez considérer que `f(n)` doit être calculé en (largement) moins d'une minute. QuestionÉcrire une fonction `f` et une fonction `f(10**6)` prenant en argument un entier naturel `somme1(n)` et renvoyant respectivement

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

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

Au besoin, on introduira des fonctions auxiliaires. 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 Écrire une fonction `somme2(n)` qui, à un entier naturel `n` et un flottant `valeur_u(n,u0)`, renvoie u_n.

Question Écrire une fonction `n` qui, à deux flottants `u0` et `approche_u(eps,u0)`, renvoie le plus petit rang $n\in\mathbb{N}$ tel que $|u_n-1|\leq$ `eps`. Question Écrire une fonction `eps` calculant $\binom{n}{p}$ pour deux entiers naturels n,p vérifiant $0\leq p\leq n$, en utilisant la formule :

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

On veillera à ce que le résultat donné par la fonction `u0` soit d'un type convenable.

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

1. C'est-à-dire, comme l'indique l'étymologie, une autre fonction dont le but sera de vous aider à répondre à la question.

Question Justifier que la fonction écrite donne bien le bon résultat, notamment à l'aide d'un invariant de boucle. Définir la fonction f qui à x associe $\begin{cases} 2 & \text{si} \quad x \in [-4,-2] \\ -x & \text{si} \quad x \in [-2,0] \\ 0 & \text{si} \quad x \in [0,4] \end{cases}$ Ecrire une fonction calculant le produit des entiers impairs de 1 à $2n+1$. Soit $(a,b,c) \in \mathbb{R}^2$, $a \neq 0$.

Ecrire 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. Écrire une fonction `comb(n,p)` qui prend en argument un entier naturel `comb` et qui renvoie la somme des chiffres de `somme_chiffres(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. Pour tout $n \in \mathbb{N}^*$, on définit

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

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

`n`

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

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

QuestionSi `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.

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 Dresser un tableau de valeurs décrivant les valeurs des variables `n` et `i` en entrée des trois premiers tours de la boucle `while` de la fonction `k`.

```
def mystere(L) :
    """Précondition : L est une li
    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
```

On pourra au besoin faire intervenir les variables `p` et `mystere(a,b)`.

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

Question Donner un variant de boucle pour la boucle `while` de la fonction `b`. On justifiera la réponse.

Question 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.