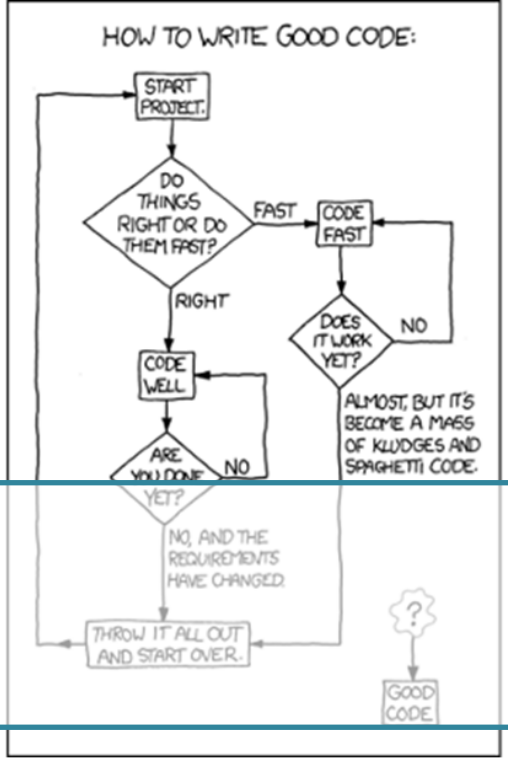

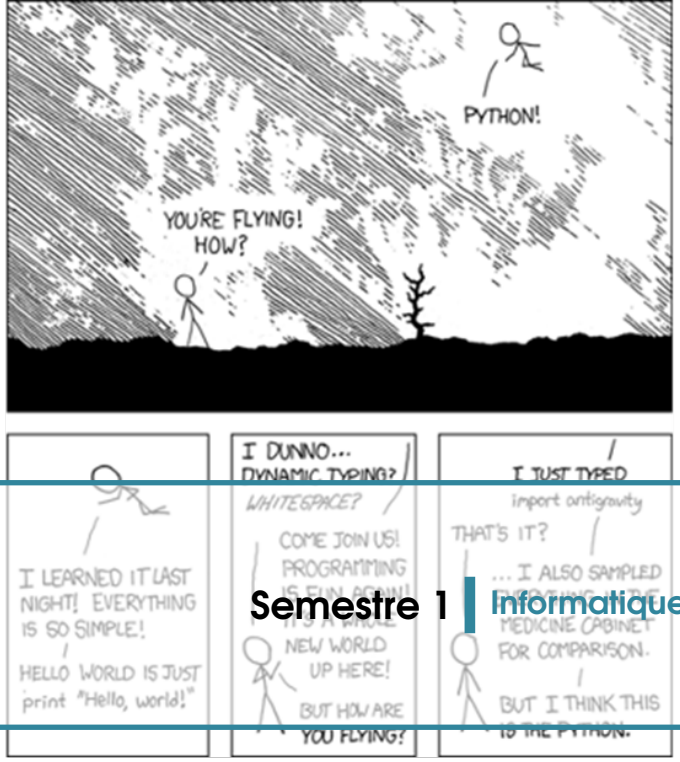

Thèmes d'étude

1 Bases de Python

Exercice 1 –

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

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

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

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

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

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

Question 4 *Évaluer les expressions suivantes.*

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

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

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

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

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

pour avoir accès à toutes ces fonctions.

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

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

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

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

Et celles-ci ?

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

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

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

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

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

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

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

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

Exercice 2 –

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

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

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

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

Exercice 3 –

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

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

Question 2 *Calculer cette suite d'expressions.*

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

Sur ce modèle, obtenir de manière synthétique :

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

Exercice 4 –

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Exercice 7 –

Question 1

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

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

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

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

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

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

Exercice 8 –

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

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

Exercice 9 –

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

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

Exercice 10 –

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

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

Exercice 11 –

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

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

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

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

```
5 in ex
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.

Modéliser

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

Exercice 19 –

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

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

Exercice 20 –

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

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

Exercice 21 –

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

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

Exercice 22 –

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

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

Exercice 23 –

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

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

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

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

Exercice 24 –

Question 1

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

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

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

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

Exercice 25 –

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

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

Exercice 26 –

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

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

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

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

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

Exercice 27 –

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

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

Exercice 28 –

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

Exercice 29 –

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

Exercice 30 –

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

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

Exercice 31 –

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

Exercice 32 –

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

Exercice 33 –

Question 1 *Écrire une fonction* `lettre` *qui prend en argument un entier* `i` *et renvoie la* $\mathtt{i}^e$ *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")
```

Modéliser

```
else:
    if n >= 50:
        print("Trop grand")
    else:
        print("Juste comme il faut")
```

Exercice 41 –

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

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

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

Exercice 42 –

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

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

Exercice 43 –

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

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

Exercice 44 –

Décrire ce que fait cette suite d'instructions.

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

Et celle-ci ?

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

Exercice 45 –

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

Exercice 46 –

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

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

Exercice 47 –

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

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

Exercice 48 –

Question 1

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

Exercice 49 –

Expliquer et justifier ce que fait la fonction suivante.

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

Exercice 50 –

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

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

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

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

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

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

Exercice 51 –

Question 1

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

Exercice 52 –

Question 1

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

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

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

Exercice 53 –

Question 1 *U*

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

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

Exercice 55 –

Question 1

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

Exercice 56 –

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

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

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

Exercice 57 –

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

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

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

Question 1

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

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

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

Question 2
Vérifier que vous pouvez calculer v_n pour de grandes valeurs de n.

Exercice 58 –

Question 1
Écrire une fonction `somme1(n)` et une fonction `somme2(n)` prenant en argument un entier naturel `n` et renvoyant respectivement

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

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

Au besoin, on introduira des fonctions auxiliaires.

Exercice 59 –

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

Question 1
Écrire une fonction `valeur_u(n,u0)` qui, à un entier naturel `n` et un flottant `u0`, renvoie u_n.

Question 2
Écrire une fonction `approche_u(eps,u0)` qui, à deux flottants `eps` et `u0`, renvoie le plus petit rang $n \in \mathbb{N}$ tel que $|u_n - 1| \leq$ `eps`.

Exercice 60 –

Question 1
Écrire une fonction `comb(n,p)` calculant $\binom{n}{p}$ pour deux entiers naturels n, p vérifiant $0 \leq p \leq n$, en utilisant la formule :

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

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

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

Question 2
Justifier que la fonction écrite donne bien le bon résultat, notamment à l'aide d'un invariant de boucle.

Exercice 61 –

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

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

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

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

Exercice 62 –

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

Exercice 63 –

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

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

Modifier pour introduire le cas de la racine double.

Exercice 64 –

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

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

Exercice 65 –

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

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

Exercice 66 –

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

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

Question 1
Écrire une fonction `H_depasse(M)` prenant en entrée un nombre `M` et renvoyant en sortie le plus petit entier naturel non nul n vérifiant $H_n \geq$ `M`.

Exercice 67 –

On considère la fonction suivante.

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

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

Question 2
Donner un variant de boucle pour la boucle `while` de la fonction `mystere`. Que peut-on en déduire ?

Question 3
Si `L` est une liste de nombres, que fait `mystere(L)` ? Le justifier, notamment à l'aide des questions précédentes

(vous pourrez cependant écrire un ou plusieurs autres invariants, au besoin).

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)` per-

Modéliser

mettant de tracer en fonction du numéro de la mensualité la dette restante (ou le capital restant dû) jusqu'à ce que le prêt soit remboursé. Cette fonction permettra également de tracer en fonction du numéro de la mensualité le montant de l'intérêt versé à la banque.

2 Algorithmique

Exercice 71 –

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

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

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

On obtient après un lissage : 1 2.5 4 5

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

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

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

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

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

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

Exercice 72 –

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

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

Ce que doit faire votre programme :

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

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

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

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

Exercice 73 –

Un marchand de légumes très maniaque souhaite ranger ses petits pois en les regroupant en boîtes de telle sorte que chaque boîte contienne un nombre factoriel de petits pois. On rappelle qu'un nombre est factoriel s'il est de la forme $1, 1\times2, 1\times2\times3, 1\times2\times3\times4\cdots$ et qu'on les note sous la forme suivante :

$$n! = 1\times2\times3\times\cdots\times(n-1)\times n$$

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 $2\times3! + 2\times2! + 1\times1! = 12+4+1 = 17$.

D'une manière générale, s'il a `nbp` petits pois, il doit trouver une suite `a_1`, `a_2`$\cdots$, `a_p` d'entiers positifs ou nuls avec `a_p`> 0 et telle que : `nbp=a_1 x 1!+a_2 x 2!+`$\cdots$`+a_p x p!` avec `a_1`+$\cdots$+`a_p` minimal.

Remarque mathématique : si à chaque étape on cherche le plus grand entier `k` possible tel que `k!` soit inférieur au nombre de petits pois restant, on est sûrs d'obtenir la décomposition optimale : en termes informatiques, on dit que l'algorithme *glouton* est optimal.

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

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

Exercice 74 –

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

Modéliser

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

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

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

EXEMPLE 1 :

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

EXEMPLE 2 :

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

Exercice 75 –

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

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

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

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

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

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

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

Exercice 76 –

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

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

Exercice 77 –

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

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

Question 1

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

Question 2

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

Exercice 78 –

Question 1

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

Question 2

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

Question 3

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

Exercice 79 –

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

chaque bloc, on effectue un « vote » et l'on considère la valeur majoritaire.

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

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

sera codé en

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

Imaginons qu'après transmission, le tableau reçu soit

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

On le décode alors en

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

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

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

Exercice 80 –

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

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

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

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

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

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

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

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

Question 3
Complexité ou invariants ?

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

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

Par exemple [...]

On considère le code suivant.

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

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

Question 6
Donner l'invariant pour la boucle for.

Question 7
Complexité.

Exercice 81 –

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

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

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

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

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

Question 1

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

Exercice 91 –

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

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

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

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

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

Exercice 92 –

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

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

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

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

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

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

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

Exercice 93 –

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

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

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

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