TP n° 02 – Lecture et écriture des premiers alogorithmes

Exercice 1. Assigner une valeur à une variable.
Au départ, on a assigné à deux variables les valeurs suivantes :

```
a=2
b=3
```

L'objectif est d'échanger les valeurs des deux variables. Pour chaque exemple, prédire la valeur finale de chaques variables et vérifier avec Python.
Certains codes renvoient un message d'erreur. Il faut les lire et analyser le problème.

```
a=b
b=a
```

```
a-1=b
```

```
a=x
a=b
b=x
```

```
x=a
a=b
b=x
```

```
(a,b)=(b,a)
```

Exercice 2. De l'importance de l'indentation
On considère les deux algorithmes suivants :

```
if a>=5:
   a=a-2
if a<4:
   a=a+5
```

Algorithme n°1

```
if a>=5:
   a=a-2
   if a<4:
      a=a+5
```

Algorithme n°2

Dans les deux cas, déterminer la valeur finale de `a` pour `a` $= 0,\ 1,\ 2, \cdots 10$.

Exercice 3. Quelle est la différence entre les deux fonctions suivantes ? Déterminer une valeur a telle que $f(a) \neq g(a)$.

```
def f(a):
    if a<-10:
         b=a-2
    elif a>15:
         b=2*a
    else :
         b=a
    return(b)
```

```
def g(a):
    if a<-10:
         b=a-2
    if a>15:
         b=2*a
    else :
         b=a
    return(b)
```

Exercice 4. Soit $f : \mathbb{R} \to \mathbb{R}$ la fonction définie par :

$$f : \quad x \quad \longmapsto \quad \begin{cases} 2x & \text{si } x \leq 0 \\ x+1 & \text{si } 0 < x \leq 1 \\ x^5 & \text{si } x > 1 \end{cases}$$

1. Programmer la fonction f en utilisant `if`, `elif`, `else`.
2. Programmer à nouveau la fonction f sans utiliser `elif` mais en utilisant deux tests `if` imbriqués. On commencera par faire l'organigramme correspondant.

Exercice 5. Dans le programme ci-dessous, `a` et `b` sont des entiers positifs.

```
while a>=b:
   a=a-b
```

1. Partant de `a=17` et `b=4`, notez à chaque étape la valeur de `a` et de `b`. Quelle est la valeur finale de `a` ?
2. En général, que vaut `a` à la fin du programme ?

Remarque. Lorsqu'on travaille avec des entiers naturels, la fonction `a//b` renvoie le quotient de la division euclidienne de `a` par `b` et `a%b` renvoie le reste de cette division euclidienne.

Exercice 6. Premières boucles for.
Recopier le programme suivant et exécuter-le.

```
for i in range(10):
    print(i)
```

1. Modifier le programme pour qu'il affiche les nombres de 0 à 20 inclus.
2. Modifier le programme pour qu'il affiche les nombres de 0 à 20 dans l'ordre décroissant.
3. Modifier le programme pour qu'il affiche les nombres de 0 à 20 qui sont pairs.
4. Dans le programme, remplacer `range(10)` par `range(2,40,3)`. A quoi correspondent les trois paramètres de la fonction `range` ?
5. Refaites la question 3 en utilisant la fonction `range` avec trois paramètres.

Remarque. Sous l'environnement IDLE, vous pouvez obtenir des informations sur un objet Python. Dans l'interpréteur intéractif, par exemple taper `help(range)` puis Entrée. Il apparaît soit directement un texte d'aide, soit une boîte sur laquelle il est possible de double-cliquer pour dérouler un texte d'aide. La partie utile pour vous est souvent au début.
Une autre méthode pour les fonctions : une info-bulle contenant aide minimale rappelant la syntaxe de la fonction apparaît automatiquement après la saisie de la parenthèse gauche après une fonction.

Exercice 7. Premiers termes d'une suite.
On considère le programme suivant :

```
u=0
for i in range(10):
    u=2*u+3
```

1. Faire tourner le programme.
2. Le programme calcule les dix premiers termes d'une suite définie par récurrence. Modifier le programme pour qu'il affiche tous les termes calculés.
3. Compléter la définition de la suite :
$$\begin{cases} u_0 = ... \\ u_{n+1} = ... \end{cases}$$
4. On considère maintenant la suite de Fibonacci définie par :
$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ \forall n \in \mathbb{N}, \quad F_{n+2} = F_{n+1} + F_n \end{cases}$$
Ecrire un programme qui permet de calculer les dix premiers termes de la suite $(F_n)_{n\in\mathbb{N}}$.

Exercice 8. Dans cet exercice, `n` est une variable déjà initialisée, entière.

```
while n>0:
    r=n%10
    n=n//10
```

1. A chaque étape, que vaut la variable `r` ?

2. Créer une fonction `somme` qui prend comme entrée un entier n et renvoie la somme des chiffres de n.

Exercice 9. On considère le programme suivant :

```
import random

nombre=random.randrange(10)                          # la variable 'nombre' est un entier
                                                     # pris au hasard entre 0 et 9
essai=int(input('entrer une valeur entre 0 et 9 '))  # la variable 'essai' est un nombre
                                                     # entré par l'utilisateur
while essai!=nombre:
   essai=int(input('entrer une nouvelle valeur '))
print('Vous avez gagné')
```

1. Recopier le programme dans un fichier Python (sans les commentaires)
2. Que fait le programme ?
3. Modifier le programme pour qu'il affiche le nombre d'essais qui ont été nécessaires pour trouver le nombre.

Exercice 10. Un nombre $n \in \mathbb{N}$ est premier si il est supérieur à 2 et si ses seuls diviseurs sont 1 et n.

1. Ecrire une fonction `premier` qui prend un entier $n \in \mathbb{N}$ comme argument et renvoie le booléen `True` si n est un nombre premier et `False` sinon.
   On commencera par déterminer quelle type de boucle est adaptée à ce programme.
2. Deux nombres premiers p et q sont dits "jumeaux" si ils ne diffèrent que de deux. Par exemple, 3 et 5 sont des nombres premiers jumeaux. Afficher les couples de nombres premiers jumeaux inférieurs à 100. [1]

Exercice 11. Facultatif : Crible d'Eratosthène.
Le crible d'Eratosthène est un algorithme qui étant donné un entier n renvoie les nombres premiers compris entre 2 et n mais de façon plus efficace que la méthode naïve de l'exercice précédent. Il fonctionne de la manière suivante :

— on inscrit tous les entiers entre 2 et n.
— 2 est premier. On stocke cette valeur et on élimine tous les multiples de 2.
— on prend le premier entier qui vient après 2 qui n'a pas été éliminé : c'est 3. On stocke cette valeur et on élimine tous les multiples de 3.
— on réitère l'opération jusqu' à n.

1. Testez l'algorithme à la main pour déterminer les nombres premiers entre 2 et 14.
2. En Python, l'algorithme est codé par le programme suivant. La liste `premier` contient la réponse.

```
premiers=[]
nombres = []
for i in range(2,n+1):
        nombres.append(True)
# nombres=[True,True,...]
#On y stockera l'information suivante :
#True : le nombre est premier, False : il ne l'est pas
for i in range(2,n+1):              # i parcourt les entiers de 2 à n
        if nombres[i-2]==True:
                premiers.append(i)
# si i est marqué comme True,
```

1. Informatiquement, on arrive à trouver des couples de nombres jumeaux jusqu'à $10^{400\,000}$. Mais on ne sait toujours pas démontrer si il en existe une infinité ou non.

```
12  # c'est un nombre premier : on le stocke dans la liste 'premier'
13                  for j in range(2*i,n+1,i):
14                          nombres[j-2] = False
15  # les multiples de i qui sont compris entre 2i et n
16  # sont alors marqués comme False
```

3. Vérifier que l'algorithme donne les mêmes valeurs que celles calculées en 1) pour `n`=14.
4. Dans cet algorithme, `i` parcourt tous les entiers entre 2 et n. Or, si i n'est pas premier, alors l'un de ces facteurs est au moins inférieur à $\sqrt{n}$[2]. Donc, dans l'algorithme, à partir de `i`= $\sqrt{n}$ (ou plutôt l'entier directement supérieur à $\sqrt{n}$), les nombres non premiers sont déjà marqués `False`. Modifier le programme pour qu'il affiche encore les nombres premiers inférieurs à n mais en évitant que `i` parcourt les entiers supérieurs à $\sqrt{n}$.

2. On suppose que $i \leq n$ n'est pas premier. Il se décompose en $i = pq$. Si $p > \sqrt{n}$ et $q > \sqrt{n}$, alors $i > n$. On dépasse alors n.

Correction TP n° 02 – Lecture et écriture des premiers alogorithmes

Solution 1.
1. `a=3` et `b=3`
2. message d'erreur : ne peut pas assigner avec un opérateur
3. message d'erreur : `x` n'est pas définie
4. `a=3` et `b=2`. On a échangé les valeurs.
5. `a=3` et `b=2`. On a échangé les valeurs.

Solution 2. Algorithme n°1 :

Valeur initiale de a	0	1	2	3	4	5	6	7	8	9	10
Valeur finale de a	5	6	7	8	4	8	4	5	6	7	8

Algorithme n°2 :

Valeur initiale de a	0	1	2	3	4	5	6	7	8	9	10
Valeur finale de a	0	1	2	3	4	8	4	5	6	7	8

Solution 3. Dans le premier cas, le test `else` est vérifié pour `a` qui n'est ni dans $]-\infty,-10[$, ni dans $]15,+\infty[$. Donc, `a` vérifie ce test pour $a \in [-10,15]$.
Dans le deuxième cas, `else` est le contraire du dernier `if`, donc le test est vérifié pour `a` qui n'est pas dans $]15,+\infty[$. Donc, `a` vérifie ce test pour $a \in]-\infty,15]$.
Par exemple, $f(-11)$ renvoie -13 mais $g(-11)$ renvoie -11.

Solution 4.
1.
```
def f(x):
    if x <= 0 :
        y = 2*x
    elif  0 < x <= 1 :
        y = x+1
    else :
        y = x**5
    return(y)
```

D'autres solutions sont possibles, comme :

```
def f(x):
    if x <= 0 :
        y = 2*x
    elif x > 1 :
        y = x**5
    else :
        y = x+1
    return(y)
```

2.
```
def f(x):
    if x<=0 :
        y = 2*x
    else :
        if x <= 1 :
            y = x+1
         else :
            y = x**
    return(y)
```

Solution 5. 1.

Etape	1	2	3	4	
Valeur de `a` avant	17	13	9	5	1
Le critère est-il vérifié ?	oui	oui	oui	oui	non
Valeur de `a` après	13	9	5	1	

La valeur finale de `a` est 1.

2. `a%b`

Solution 6. 1.

```
for i in range(21):
    print(i)
```

2.
```
for i in range(21):
    print(20-i)
```

3.
```
for i in range(11):
    print(2*i)
```

Solution 7.
```
for i in range(0,21,2):
    print(i)
```

Solution 8. 1. .

2.
```
u=0
for i in range(9):
    u=2*u+3
```

3. $\begin{cases} u_0 = 0 \\ u_{n+1} = 2u_n + 3 \end{cases}$

4.
```
u=0
v=1
for i in range(10):
    (u,v)=(v,u+v)
print(v)
```

Solution 9. 1. La variable `r` vaut successivement tous les chiffres du nombre `n`.

2.
```
def somme(n):
        s=0                 # on initialise la somme a zero
        while n>0:
                r=n%10          # r est le dernier chiffre de n
                    s=s+r           # on ajoute r a la somme
                n=n//10         # on recommence avec n auquel on a retiré le dernier chiffre
        return(s)
```

Solution 10.

```
import random

nombre=random.randrange(10)                          # la variable 'nombre' est un entier
                                                     # pris au hasard entre 0 et 9
compteur=1                                           # on initialise le compteur a 1
essai=input('entrer une valeur entre 0 et 9 ')       # la variable 'essai' est un nombre
                                                     # entré par l'utilisateur
while essai!=nombre:
   essai=input('entrer une nouvelle valeur ')
   compteur=compteur+1                               # a chaque coups, compteur augmente de 1
print('Vous avez gagné en ',compteur,' coups')
```

Solution 11. 1.

```
def premier(n):
    if n==0 or n==1:
        return(False)
    else:
        i=2
        while i<=n//2 and n%i!=0:
            i=i+1
        if i==n//2+1:
            return(True)
        else:
            return(False)
```

2.

```
for i in range(100):
    if premier(i) and premier(i+2):
        print(i,i+2)
```

Solution 12. 1. Nombres premiers entre 2 et 14 :
$2, 3, \underline{4}, 5, \underline{6}, 7, \underline{8}, \underline{9}, \underline{10}, 11, \underline{12}, 13, \underline{14}$.

2.

```
n=100
premiers=[]
nombres = []
N=int(sqrt(n))+1                    # N est l'entier strictement superieur à racine de n

for i in range(2,n+1):
    nombres.append(True)            # nombres=[True,True,...] : on y stocke l'information
                                    # True : le nombre est premier ou False : il ne l'est pas
for i in range(2,N+1):              # i parcourt les entiers de 2 à N
    if nombres[i-2]==True:          # si i n'est pas False,
        premiers.append(i)          # c'est un nombre premier, on le stocke dans la liste
        for j in range(2*i,n+1,i):  # les multiples de i entre 2i et n
            nombres[j-2] = False    # sont alors marqués comme False


for i in range(N+1,n+1):            # on ajoute a la liste 'premiers' les nombres superieurs a
    if nombres[i-2]==True:          # N qui sont marqués True
        premiers.append(i)
print(premiers)
```