TD, Programmation (1) : Corrigé.

1 Algorithmique sans programmation

Exercice 1. *Déplacement d'un robot dans le plan.* Un robot, initialement situé au point $(0,0)$ du plan, peut se déplacer dans les directions cardinales d'une unité (exemple : déplace-toi d'un cran vers le nord). Le but du jeu est de le positionner à une position victorieuse (x,y) particulière, avec x et y entiers.

1. On suppose que la position victorieuse est sur l'axe des abscisses, à une abscisse strictement positive (donc $(x,0)$ avec $x>0$). Décrire comment placer le robot sur la position victorieuse.
2. Même chose à une position (x,y) quelconque.
3. On suppose maintenant que la position victorieuse est de la forme $(x,0)$ avec $x>0$, mais x est inconnu. On peut interroger le robot sur « es-tu en position victorieuse ? ». Décrire comment placer le robot à la bonne position.
4. Même chose avec une position (x,y) quelconque et inconnue (on ne formalisera pas forcément, mais on décrira une stratégie permettant de trouver la position victorieuse).

Corrigé. On décrit brièvement les opérations à effectuer ici. On pourrait tout écrire en pseudo-code.

1. Il suffit de réaliser x fois l'opération « se déplacer d'un cran à droite ».
2. Si $x>0$, on réalise x fois l'opération « se déplacer d'un cran à droite », sinon on réalise $-x$ fois l'opération « se déplacer d'un cran à gauche ». Ensuite on fait de même sur les ordonnées, vers le haut ou vers le bas.
3. On démarre avec $x=0$, et tant que $(x,0)$ n'est pas victorieuse, on réalise l'opération $x \leftarrow x+1$.
4. L'idée est de choisir une énumération de toutes les points entiers du plan. On peut par exemple construire une « spirale » partant de $(0,0)$, passant par tous les points à coordonnées entières du plan. Tant que la position n'est pas victorieuse, passer au point suivant. (C'est un bon exercice que de décrire entièrement l'algorithme suivi !).

2 Instructions conditionnelles

Exercice 2. On suppose deux variables `taille` et `poids` déja affectées. L'indice de masse corporelle (abrégé IMC, égal à $\frac{\text{poids}}{\text{taille}^2}$) d'un individu donne une indication sur sa santé. Calculer l'IMC que vous stockerez dans une variable `imc`, et indiquer par un message à l'écran si l'individu est en surpoids (IMC >25) ou en sous-poids (IMC <18).

Corrigé.

```
imc = poids / taille**2
if imc>25:
    print("surpoids")
elif imc<18:
    print("sous-poids")
```

Remarque : pas de `else` s'il n'y en a pas besoin.

Exercice 3. On suppose que `a` et `b` sont des variables booléennes. Indiquer les instructions à effectuer pour calculer `not a`, `a and b` et `a or b`, sans utiliser ces opérateurs mais à l'aide d'instructions conditionnelles (le moins possible !). Stocker les résultats dans `non_a`, `a_et_b` et `a_ou_b`.

Corrigé. Le plus court :

```
if a:
    non_a = False
else:
    non_a = True

if a:
    a_et_b = b
else:
    a_et_b=False

if a:
    a_ou_b = True
else:
    a_ou_b = b
```

Exercice 4. On suppose que `a`, `b` et `c` sont des variables contenant trois nombres. Stocker dans la variable `d` le plus petit, sans utiliser la fonction `min` de Python.

Corrigé.

```
if a<b:
    if a<c:
        d=a
    else:
        d=c
else:
    if b<c:
        d=b
    else:
        d=c
```

Exercice 5. *Suite de l'exercice précédent.* Calculer le triplet `t` contenant les trois nombres précédents, mais triés dans l'ordre croissant. Il y a 6 possibilités.

Corrigé.

```
if a<b:
    if a<c:
        if b<c:
            t=a,b,c
        else:
            t=a,c,b
    else:
        t=c,a,b
else:
    if b<c:
        if a<c:
            t=b,a,c
        else:
            t=b,c,a
    else:
        t=c,b,a
```

3 Boucles

Exercice 6. À l'aide de boucles `for`, calculer les sommes suivantes :

$$\sum_{i=0}^{100} i^2 \qquad \sum_{i=0}^{100}\sum_{j=0}^{100} ij \qquad \sum_{i=0}^{100}\sum_{j=i}^{100} ij \qquad \sum_{i=0}^{100}\sum_{j=0}^{i} ij$$

Corrigé. Voici les 4 sommes :

```
s1=0
for i in range(101):
    s1+=i**2
```

```
s2=0
for i in range(101):
    for j in range(101):
        s2+=i*j
```

```
s3=0
for i in range(101):
    for j in range(i,101):
        s3+=i*j
```

```
s4=0
for i in range(101):
    for j in range(i+1):
        s4+=i*j
```

Exercice 7. Que vaut `s` après la boucle suivante, en supposant les variables `x` et `N` affectées (`N` contient un entier positif) ?

```
s=0
y=1
for i in range(N):
    s=s+y
    y=y*x
```

Corrigé. s vaut $\sum_{i=0}^{N-1} x^i$.

Exercice 8. On suppose les variables `a` et `b` affectées, elles contiennent des entiers positifs. Stocker dans `c` le produit ab, en n'effectuant que des additions et des soustractions. Le faire avec une boucle `for` et une boucle `while`.

Corrigé. Par exemple :

```
c=0
for i in range(a):
    c+=b
```

```
c=0
while a>0:
    c+=b
    a-=1
```

Exercice 9. On peut montrer que la suite récurrence $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 1$ et $u_{n+1} = \sin(u_n)$ pour $n \geq 0$ est une suite décroissante qui tend vers 0 lorsque $n \to +\infty$. On se donne $\varepsilon > 0$ stocké dans dans la variable `eps`. Donner une suite d'instructions permettant de trouver l'indice n du premier terme u_n vérifiant $u_n < \varepsilon$. On supposera la fonction `sin` importée du module `math`.

Corrigé.

```
u=1
n=0
while u>=eps:
    u=sin(u)
    n+=1
```

Exercice 10. Un entier $N \geq 2$ est dit premier s'il n'est divisible par aucun entier entre 2 et $N-1$. Un test usuel de primalité consiste à tester la divisibilité par tous les entiers entre 2 et $\sqrt{N}$ (en effet, si un entier est non premier, son plus petit diviseur non trivial est inférieur ou égal à $\sqrt{N}$, le pire cas étant celui d'un carré de nombre premier). En supposant la variable N affectée, écrire une suite d'instructions permettant de décider si N est premier. On pourra stocker dans une variable `estpremier` un booléen. Contrainte : on travaillera uniquement sur des entiers, pas de flottants ici.

Corrigé.

```
estpremier=True
b=2
while b*b<=N:
    if N%b==0:
        estpremier=False
    b+=1
```

Exercice 11. La suite de Fibonacci est définie par $F_0 = 0$, $F_1 = 1$ et pour tout $i \geq 2$, $F_i = F_{i-2} + F_{i-1}$. En utilisant deux variables et une boucle, calculer F_{100}.

Corrigé. On utilise deux variables, qui en haut du corps de la boucle contiennent F_{i-1} (`a`) et F_{i-2} (`b`), ainsi qu'une variable intermédiaire `c` :

```
a, b=1, 0
for i in range(2,101):
    c=b
    b=a
    a=a+c
    #ici a contient F_i
#ici a contient F_100
```

Exercice 12. Écrire un code permettant d'afficher à l'écran tous les triplets pythagoriciens (triplets d'entiers (a, b, c) tels que $a^2 + b^2 = c^2$), tels que $1 \leq a \leq b < c \leq 1000$.

Corrigé. Ce code fonctionne, mais n'est pas le plus efficace :

```
for a in range(1,1000):
    for b in range(a,1000):
        for c in range(b+1,1001):
            if a**2+b**2 == c**2:
                print(a,b,c)
```

Remarque : il vaut mieux vérifier que $\sqrt{a^2+b^2}$ est un entier, cela prend moins de temps !

4 Listes

Exercice 13. *Palindrome.* On suppose la variable `L` affectée. On dit que `L` est un palindrome si ses éléments sont les mêmes si `L` est lue en sens inverse. Par exemple `[0, 1, 1, 0]` ou `[0, 1, 2, 1, 3, 1, 2, 1, 0]` sont des palindromes.

1. Écrire un script permettant d'affecter à `Linv` la liste des éléments de `L` lus en sens inverse, sans toucher à `L`.
2. En déduire comment affecter à la variable `est_palindrome` un booléen indiquant si `L` est un palindrome.
3. Reprendre la question précédente sans faire usage d'une liste intermédiaire.

Corrigé.

1.
```
Linv=[]
for i in range(len(L)-1, -1, -1):
    Linv.append(L[i])
```

2.
```
est_palindrome = Linv == L
```

3.
```
est_palindrome = True
for i in range(len(L)//2):
    if L[i] != L[len(L)-1-i]:
        est_palindrome = False
```

Exercice 14. *Liste de bits.* On se donne une liste `L` constituée uniquement de zéros et de uns. Écrire un script stockant dans la variable `Ltriee` une liste contenant les mêmes éléments que `L`, mais les zéros étant en premier.

Corrigé. Il suffit de compter le nombre de zéros, on en déduit le nombre de uns et on peut construire `Ltriee` :

```
c=0
for i in range(len(L)):
    if L[i]==0:
        c+=1
Ltriee = [0]*c+[1]*(len(L)-c)
```

Exercice 15. *Présence d'un élément dans une liste.* On se donne une liste `L` et un élément `x`. Écrire un code permettant de stocker dans la variable `est_present` un booléen indiquant si `x` est présent dans `L`.

Corrigé.

```
est_present = False
for i in range(len(L)):
    if L[i]==x:
        est_present=True
```