TD : Algorithmique et programmation (1)

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).

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).

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`.

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.

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.

3 Boucles

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

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

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
```

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`.

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`.

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.

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}.

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 $1 \leq a \leq b < c \leq 1000$.

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.

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.

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`.