

Exercices : algorithmique avec Python

Cette fiche propose une traduction des algorithmes de la feuille de cours précédente en langage Python.

I/ Instructions conditionnelles

Exemple :

On considère l'algorithme ci-dessous :

Variables	A et B sont des nombres réels
Entrées	Saisir A et B
Traitement et sortie	Si $A < B$ Alors afficher A Sinon afficher B Fin Si

Traduction en langage Python

```
# Fonction permettant de renvoyer
# la plus petite des deux valeurs
def comparer(a,b) :
    if a < b :
        return a
    else :
        return b

# Saisie des valeurs et conversions
# en nombre flottant (réels)
A = float(input("Saisir la variable A : "))
B = float(input("Saisir la variable B : "))

print("La plus petite valeur est",comparer(A,B))
```

```
Saisir la variable A : 5
Saisir la variable B : -5.4
La plus petite valeur est -5.4
```

Remarque : La commande `input()` permet de **demandeur une valeur à l'utilisateur**. Attention, elle est systématiquement traduite en chaîne de caractères, il faut donc la **convertir** dans le type souhaité (`int(valeur)` pour un nombre entier et `float(valeur)` pour un nombre réel).

Question : Traduire sur Jupyter en langage Python les algorithmes des exercices 1 et 2. On pensera à utiliser une fonction, le programme n'en sera que plus lisible.

Aller plus loin : même question pour l'exercice 3.

Exercice 1 :

La directrice d'un commerce de reprographie a créé un algorithme permettant de déterminer le montant payé par un client à partir du nombre de photocopies effectuées.

Variables	N est un entier, P est un nombre réel
Entrée	Saisir N
Traitement	Si $A < 30$ Alors P prend la valeur $A \times 0,2$ Sinon P prend la valeur $6 + (A - 30) \times 0,1$ Fin Si
Sortie	Afficher P

```
def calcul_prix(n) :
    if n < 30 :
        return n*0.2
    else :
        return 6 + (n-30)*0.1

A = int(input("Saisir le nombre de photocopies : "))

print("Le prix à payer est",calcul_prix(A))

Saisir le nombre de photocopies : 20
Le prix à payer est 4.0
```

Exercice 2 :

Voici un algorithme :

Variables	A, B, C et D sont des réels
Entrées	Saisir A et B
Traitement	C prend la valeur A - B Si C ≤ 0 Alors Affecter à D la valeur B - A Sinon Affecter à D la valeur A - B Fin Si
Sortie	Afficher D

```
def valeur_absolue(a,b) :  
    c = a - b  
    if c <= 0 :  
        return -c  
    else :  
        return c  
  
# Saisie des valeurs et conversions  
# en nombre flottant (réels)  
A = float(input("Saisir la variable A : "))  
B = float(input("Saisir la variable B : "))  
  
print("La valeur absolue de A-B est",valeur_absolue(A,B))  
  
Saisir la variable A : -6  
Saisir la variable B : -2  
La valeur absolue de A-B est 4.0
```

Remarque : la variable D n'est pas nécessaire, si on veut l'utiliser, il faut l'initialiser à 0 juste sous l'instruction $c = a - b$ sinon elle ne sera pas définie dans la condition !

II/ Boucles bornées

Exemple :

Traitement et sortie	Pour i variant de 1 à 5 faire Afficher « bonjour ! » Fin Pour
-----------------------------	---

```
for i in range(1,6) :  
    print("Bonjour")
```

Bonjour
Bonjour
Bonjour
Bonjour
Bonjour

Remarque : dans le programme ci-dessus, la variable i va varier entre **1 inclus et 6 exclu**.

Pour répéter 5 fois une instruction, les scripts suivants sont identiques :

```
for i in range(0,5) :  
    print("Bonjour")
```

Bonjour
Bonjour
Bonjour
Bonjour
Bonjour

Ici, i varie de 0 à 4 inclus.

```
for i in range(5) :  
    print("Bonjour")
```

Bonjour
Bonjour
Bonjour
Bonjour
Bonjour

Même chose ici.

```
for i in range(0,10,2) :  
    print("Bonjour")
```

Bonjour
Bonjour
Bonjour
Bonjour
Bonjour

Ici, i vaut 0 ; 2 ; 4 ; 6 ; 8

Question : Traduire sur Jupyter en langage Python les algorithmes des exercices 1 à 3. On pensera à utiliser une fonction, le programme n'en sera que plus lisible.

Aller plus loin : même question pour l'exercice 4.

Exercice 1 :

Voici un algorithme :

Variables	n et S sont des entiers
Entrée	Saisir n
Initialisation	S prend la valeur 0
Traitement	Pour i variant de 1 à n faire S prend la valeur $S + i$ Fin Pour
Sortie	Afficher S

```
def somme(N) :  
    S = 0  
    # Attention au 'range'  
    for i in range(N+1) :  
        S += i  
  
    return S  
  
# Saisie de N et conversion  
# en entier  
N = int(input("Saisir la valeur : "))  
  
print("La somme des N premiers termes est : ", somme(N))
```

Saisir la valeur : 100
La somme des N premiers termes est : 5050

Exercice 2 :

Compléter la deuxième ligne de l'algorithme suivant pour qu'il affiche successivement :

a. 0, 7, 14, 21 et 28; **b.** 21, 28, 35, 42, 49, 56 et 63.

Variable	P est un entier
Traitement et sortie	Pour i variant de ... à ... faire P prend la valeur $7 \times i$ Afficher P Fin Pour

a)

```
def affiche() :  
    p = 0 # A initialiser  
    for i in range(0,5) :  
        # Affiche et sépare les termes par des espaces  
        p = 7*i  
        print(p, end= " ")  
  
affiche()
```

0 7 14 21 28

b)

```
def affiche() :  
    p = 0 # A initialiser  
    for i in range(3,10) :  
        # Affiche et sépare les termes par des espaces  
        p = 7*i  
        print(p, end= " ")  
  
affiche()
```

21 28 35 42 49 56 63

Exercice 3 :

A noter : En langage Python, a^b s'écrit $a**b$.

Exécuter l'algorithme suivant avec $n = 6$ en entrée.

Variables	E et n sont des entiers
Entrée	Saisir n
Initialisation	E prend la valeur 1
Traitement	Pour i variant de 1 à n faire E prend la valeur $E + 10^i$ Fin Pour
Sortie	Afficher E

```
def calcule() :  
    E = 1  
    for i in range(1,7) :  
        E += 10**i  
    return E  
  
print("La valeur de E est", calcule())
```

La valeur de E est 1111111

III/ Boucles non bornées

Exemple :

Compléter l'algorithme pour qu'il affiche tous les multiples entiers naturels de l'entier A strictement inférieurs à 1000.

Variables	A, K et M sont des entiers
Entrée	Saisir A
Initialisation	K prend la valeur 0 M prend la valeur 0
Traitement	Tant que faire Afficher M Affecter à K la valeur K + 1 M prend la valeur Fin Tant que

```
def multiples(A) :  
    M = A # Contient les multiples de A  
    K = 1 # Compteur  
  
    # On ne doit pas dépasser 1000  
    while M < 1000 - A :  
        M = K*A  
        # Ecrit les multiples sur une ligne  
        print(M,end = " ")  
        K += 1 # K = K + 1  
  
# Saisie de N et conversion  
# en entier  
A = int(input("Saisir le nombre : "))  
  
print("Les multiples de",A," sont : ")  
multiples(A)
```

```
Saisir le nombre : 15  
Les multiples de 15 sont :  
15 30 45 60 75 90 105 120 135 150 165 180 195 210 225 240 255 270 285 300 315 330 345 360 375 390 405 420 435 450 465 480 495 510 525 540 555 570 585 600 615 630 645 660 675 690 705 720 735 750 765 780 795 810 825 840 855 870 885 900 915 930 945 960 975 990
```

Question : Traduire sur Jupyter en langage Python les algorithmes des exercices 1 et 2. On pensera à utiliser une fonction, le programme n'en sera que plus lisible.

Aller plus loin (*) : Les parents de Sidonie ont placé sur son livret d'épargne la somme de 1000 euros en 2010. On suppose que le taux d'intérêt est de 3%.

Ecrire un programme en Python donnant l'année pour laquelle il y aura au moins 1200 euros sur le compte (on supposera qu'aucun retrait d'argent n'a été effectué).

Exercice 1 :

On considère l'algorithme suivant :

Variable	U est un entier
Entrée	Saisir U
Traitement	Tant que U > 7 faire U prend la valeur U - 7 Fin Tant que
Sortie	Afficher U

```
def calcule(u) :  
    while u > 7 :  
        u -= 7 # Equivaut à u = u - 7  
    return u  
  
U = int(input("Saisir la variable U : "))  
  
print("La variable U vaut désormais",calcule(U))
```

```
Saisir la variable U : 25  
La variable U vaut désormais 4
```

Remarque : Cette fonction calcule le reste de la division euclidienne de 25 par 7. L'équivalent en Python est l'instruction suivante, $25 \% 7$.

Ceci est très pratique pour déterminer par exemple si un nombre b est pair : il suffit de tester si $b \% 2$ vaut zéro ! Un exemple concret d'utilisation cette année avec le principe du photomaton (partie photo numérique).

Exercice 2 :

Compléter l'algorithme suivant afin qu'il donne en sortie la plus petite valeur de l'entier N pour laquelle la somme des N premiers entiers naturels dépasse 10000.

Variables	N et S sont des entiers
Initialisation	S prend la valeur 0 N prend la valeur 0
Traitement	Tant que faire N prend la valeur $N + 1$ S prend la valeur
	Fin Tant que
Sortie	Afficher N

```
def somme() :  
    S = 0 # Contient la somme  
    n = 1 # Stocke l'entier cherché  
  
    while S < 10000 - n :  
        S += n  
        n += 1  
  
    # Ne pas oublier l'ajout du dernier n dans S  
    return n, S + n  
  
# Renvoie le nombre cherché + leur somme  
nombre, somme = somme()  
print("La somme des", nombre, "premiers nombres vaut", somme)
```

La somme des 141 premiers nombres vaut 10011

Correction des « Aller plus loin »

Partie « instruction conditionnelle »

```
def prix_forfait(n_min) :  
    if n_min <= 120 :  
        return 8  
    else :  
        return 8 + (n_min-120)*0.2  
  
# Penser à convertir le forfait en minutes  
N = int(input("Saisir le nombre de minutes du forfait (en minutes) : "))  
  
print("Le montant du forfait est de", prix_forfait(N), "euros")
```

Saisir le nombre de minutes du forfait (en minutes) : 140
Le montant du forfait est de 12.0 euros

Partie « boucle bornée »

```
def somme_carres(n) :  
    s = 0  
    # Attention à bien considérer le Nième nombre  
    for i in range(n+1) :  
        s += i**2  
    return s  
  
# Penser à convertir le forfait en minutes  
N = int(input("Saisir la variable N : "))  
  
print("La somme des N premiers carrés vaut", somme_carres(N))
```

Saisir la variable N : 20
La somme des N premiers carrés vaut 2870

Partie « boucle non bornée »

```
def calcul_annee() :  
    S = 1000 # Capital de départ  
    annee = 2010 # Stocke l'année  
  
    while S < 1200 :  
        S *= 1.03  
        annee += 1  
  
    # Renvoie l'année  
    return annee  
  
# Renvoie le nombre cherché + leur somme  
an = calcul_annee()  
print("En", an, ", le capital aura dépassé 1200 euros")
```

En 2017, le capital aura dépassé 1200 euros

Remarque : Bien faire attention aux valeurs renvoyées après une boucle *while*, il faut parfois tenir compte « d'un tour en trop ».