TP 05 - Les bases en Python/Exercices

TP 05 : Les bases en Python	**Thème 4 : Langages et Programmation**
	EXERCICES

Lien Capytale

Objectif

L'objectif final est que sachiez écrire les structures de bases (affectations, instructions conditionnelles, boucles `for`, boucle `while`, fonctions) dans le langage Python.

Pour cela, une grande partie des exerciec ci-dessous consistent à écrire dans le langage Python, des algorithmes écrits en langage naturel dans la partie 1 sur *Les bases de l'algorithmique*.

> N'hésitez surtout pas à revoir le notebook de cours sur les bases de Python, faites des allers-retours entre le cours et les exercices !

I. Variables et affectations en Python

Exercice 1

Exercice n°1 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 1 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses en affichant ce qu'il faut avec Python*.

> Que valent `N` et `P` après l'exécution de cet algorithme ?

```
N ← 2
P ← 3
N ← P + 1
P ← N
```

Solution

```
N=2
P=3
N=P+1
P=N
print(f"N : {N} et P : {P}")
```

Exercice 2

Exercice n°2 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 2 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses en affichant ce qu'il faut avec Python*.

> Que vaut `B` après l'exécution de cet algorithme ?
>
> ```
> A ← 8
> B ← (A ≥ 10)
> ```

Solution

```
A = 8
B = (A >= 10)
print(B)  ## pour voir la valeur de B
```

Exercice 3

Exercice n°3 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 3 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Supposons que `N` vaut 2 ; que vaut `P` après l'exécution de cet algorithme ?
>
> ```
> P ← N mod 5
> P ← N div P
> ```

Solution

```
N = 2  # initialisation de N
P = N % 5
P = N // P
print(P)  # pour voir la valeur de P
```

II. Instructions conditionnelles : `if`, `elif` et `else`

Exercice 4

Exercice n°4 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 5 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Supposons que `A` vaut 1 au début ; que vaut-il après l'exécution de cet algorithme ?

```
si A ≥ 0
alors A ← A - 1
sinon A ← A + 1
fin si
```

Solution

```
A = 1  ## valeur intiale de A
if A >= 0:
    A = A - 1
else:
    A = A + 1
print(A)  # pour voir la valeur finale de A
```

Exercice 5

Exercice n°5 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 6 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> L'algorithme ci-dessous est-il équivalent (au sens ou une même valeur de `A` initiale induit une même valeur de `A` après leur exécution) à celui présenté dans l'exercice 5 ?

```
si A ≥ 0
alors A ← A - 1
fin si
si A < 0
alors A ← A + 1
finsi
```

Solution

```
A = 0  ## valeur initiale de test
if A >= 0:
    A = A - 1
if A < 0:
    A = A + 1
print(A)  ## pour voir la valeur finale
```

III. Boucles bornées : boucles `for`

Exercice 6

Exercice n°6 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 9 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Supposons que `A` vaut initialement 10 ; combien vaut-il après l'exécution de cet algorithme ?

```
pour i de 2 à 5 faire
    A ← A + i
fin pour
```

Solution

```
A = 10
for i in range(2, 6):  ## Attention : 6 !
    A = A + i
print(A)  ## pour voir la valeur finale de A
```

Exercice 7

Exercice n°7 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 10 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Supposons que `V` vaut 0 initialement ; combien vaut-il après l'exécution de cet algorithme ?
>
> ```
> pour I de 1 à 4 par pas de 1 faire
> si I est pair
>     alors V ← V + I*I
> fin si
> fin pour
> ```

Solution

```
V = 0
for I in range(1, 5):
    if I%2 == 0:  ## si i est divisible par 2
        V = V + I*I
print(V)  ## pour voir la valeur finale de V
```

Exercice 8

Exercice n°8 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 12 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Supposons que `A` vaut 1 initialement ; combien vaut-il après l'exécution de cet algorithme ?
>
> ```
> pour i de 0 à 3 faire
>     pour j de 0 à 2 faire
>         A ← A + i*j
>     fin pour
> fin pour
> ```

Solution

```
A = 1  # initialisation de A
for i in range(4):  ## ou range(0, 4)
    for j in range(3):
```

```
        A = A + i*j
print(A)  # pour voir la valeur finale de A
```

Exercice 9

Exercice n°9 :

Enonce

Q1 : Quelles sont les valeurs affichées dans la console lors de l'exécution de chacun des programmes ci-dessous ? *Attention, il faut répondre sans exécuter le code !* * Programme 1 :

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

* Programme 2 :

```
for k in range(2, 10):
    print(k)
```

* Programme 3 :

```
for ind in range(2, 10, 3):
    print(ind)
```

Q2 : Vérifiez en réécrivant et en exécutant chacun de ces programmes.

Solution

- Programme 1 : 0, 1, 2, ..., 9
- Programme 2 : 2, 3, 4, ..., 9
- Programme 3 : 2, 5, 8

Exercice 10

Exercice n°10 :

Enonce

On considère le programme Python suivant :

```
chaine = 'peace and love'
compteur = 0
for caractere in chaine:
    if caractere == 'e':
        compteur = compteur + 1
```

Q1 : Quelle est le rôle de ce code. Quelle est la valeur finale de la variable `compteur` ?

Réponse (à compléter) :

Q2 : Recopiez ce code et vérifiez la réponse à la question précédente.

```
# à vous de jouer !
```

Q3 : Modifiez le code pour compter le nombre de `'a'`.

```
# à vous de jouer !
```

Solution

- Q1 : Il compte le nombre d'apparition du caractère 'e' dans chaine. La variable finale de compteur est donc 3.
- Q2 : Il n'y qu'à recopier
- Q3 : Il suffit de remplacer 'e' par 'a' dans l'instruction conditionnelle à la ligne 4

IV. Boucles non bornées : boucles `while`

Exercice 11

Exercice n°11 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 13 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Supposons que `D` vaut 7 initialement ; que vaut-il après l'exécution de cet algorithme ? ``` tant que D > 3 faire D ← D - 3 fin tant que

> ```python
> # écrivez le programme Python correspondant
> ```

Solution

```
D = 7  ## valeur initiale de D
while D > 3:
    D = D - 3
print(D)  ## pour voir la valeur finale de D
```

Exercice 12

Exercice n°12 :

Enonce

Reprenez le programme Python de l'exercice précédent et ajoutez-y une variable qui compte le nombre d'itérations dans la boucle `while`. Cela pour permettra de vérifier la question 2 de l'exercice 14 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> On considère l'algorithme précédent. 1. Supposons que `D` vaut 2 après l'exécution de cet algorithme. Quelles valeurs parmi celles proposées ci-dessous pouvait-il avoir initialement ? -2 ; 2 ; 6 ; 14 ; 25. 2. Supposons que `D` vaut 25 initialement ; combien d'itérations (c'est-à-dire de passages dans la répétitive) ont lieu avant que l'exécution de cet algorithme ne se termine ?

```
# ajoutez une variable qui compte le nombre d'itératio
```

Solution

On ajoute la variable `iter` qui vaut 0 au départ et que l'on augmente de 1 à chaque tour de boucle. On remplace aussi la valeur intiale de D par 25.

```
D = 25  # valeur initiale de D
iter = 0
while D > 3:
    iter = iter + 1
    D = D - 3
print(iter, D)  # pour voir le nombre total d'itératio
```

Exercice 13

Exercice n°13 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 15 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Proposez un algorithme qui utilise une boucle Tant que qui a pour but d'afficher le quotient de la division entière d'un entier naturel A par un entier naturel B.

```
# écrivez le programme Python correspondant
```

Solution

L'idée est la suivante : avec une boucle Tant que, on retranche B à A jusqu'à obtenir un nombre < B. Le nombre de fois que l'on a pu faire cela est le quotient cherché, c'est donc le nombre de passages dans la boucle Tant que. On utilise une variable q qui compte le nombre de passages comme dans l'exercice précédent.

```
# valeurs pour tester
A = 25
B = 3
# algorithme
q = 0
while A >= B:
    q = q + 1
    A = A - B
# pour voir le quotient
print(q)
```

V. Les fonctions

Exercice 14

Exercice n°14 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 16 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Proposez le pseudo-code d'une fonction `minimum` qui renvoie le minimum des deux nombres donnés en paramètres.

```
# écrivez la fonction en Python
```

Solution

```
def minimum(a, b):
    if a < b:
        m = a
    else:
        m = b
    return m
```

ou

```
def minimum(a, b):
    if a < b:
        return a
    else:
        return b
```

Exercice 15

Exercice n°15 :

Enonce

Écrivez le programme Python correspondant à l'algorithme de l'exercice 17 sur les bases de l'algorithmique dont on rappelle l'énoncé ci-dessous. *Vous vérifiez également vos réponses avec Python*.

> Proposez le pseudo-code d'une fonction `minimum4` qui renvoie le minimum des 4 nombres donnés en paramètres. **Contrainte** : vous devez faire appel à la fonction `minimum` !

```
# écrivez la fonction en Python
```

Solution

```
def minimum4(a, b, c, d):
    m1 = minimum(a, b)
    m2 = minimum(c, d)
    m = minimum(m1, m2)
return m
```

Exercice 16

Exercice n°16 :

Enonce

Adaptez le code de l'exercice 10 pour écrire une fonction `compte_caractere(c, chaine)` qui renvoie le nombre de fois où apparaît le caractère `c` dans la chaîne de caractères `chaine` (on parle du nombre d'occurrences de `c`).

Exemple : l'appel `compte_caractere('e', 'peace and love')` doit renvoyer la valeur 3, ce que l'on écrit :

```
>>> compte_caractere('e', 'peace and love')
3
```

```
# à vous de jouer !
```

Solution

```
def compte_caractere(c, chaine):
    compteur = 0
    for caractere in chaine:
        if caractere == c:
            compteur = compteur + 1
    return compteur
```