

A. Algorithmique - Premiers algorithmes:

Définition : Un **algorithme** est une liste d'instructions à exécuter pas à pas pour accomplir une tâche précise.

Remarques

- Un algorithme doit être lisible de tous ; Son intérêt est d'être codé dans un langage informatique pour qu'une machine (ordinateur, calculatrice, etc.) puisse l'exécuter rapidement et efficacement.
- Les trois phases d'un algorithme sont :
 - l'entrée des données ;
 - le traitement des données ;
 - la sortie des résultats.

Ex 1 - On considère l'algorithme suivant :

Choisir un nombre de départ	3					
Lui ajouter 1	4					
Multiplier le résultat par 2	8					
Soustraire 3 au résultat et afficher le résultat final	5					

- 1) Recopier le tableau ci-dessus, puis :
 - a) Appliquer successivement cet algorithme à -4 , 0 , $1/3$
 - b) Déterminer les nombres à choisir au départ pour afficher à la fin 0 puis 7
- 2) Écrire un nouvel algorithme qui permette, en partant du résultat final de l'algorithme ci-dessus, de retrouver le nombre de départ.
- 3) Traduire chacun de ces deux algorithmes par une formule en fonction du nombre de départ x .
Quelle est la nature des deux fonctions trouvées ?

Ex 2 - On considère l'algorithme suivant :

Choisir un nombre
Calculer le carré de ce nombre
Multiplier par 10
Ajouter 25
Afficher le résultat

- 1) En faisant un tableau comme celui de l'exercice I), appliquer cet algorithme à 2 , puis $\sqrt{2}$
- 2) Traduire cet algorithme par une formule en fonction de x .
- 3) Pénélope affirme que si le nombre choisi au départ est un entier alors le résultat est impair.
A-t-elle raison ? Justifier.
- 4) Ulysse affirme que le résultat est toujours positif quelque soit le nombre choisi au départ.
A-t-il raison ? Justifier.

B. Le langage Python:

Les algorithmes sont traduits en programmes informatiques à partir des différents langages de programmation qui existent. Dans ce chapitre, nous utiliserons **Python** comme langage de programmation.

1. Éléments de base

a. Les types

Le type d'une variable définit l'ensemble des valeurs qui peuvent lui être affectées. Les types de base sont **int**, **bool**, **float** qui sont des types numériques et **str**.

Ce qu'il faut savoir :

- ✓ Le type **int**, abréviation de integer, est utilisé pour représenter les **nombres entiers**.
- ✓ Le type **bool** permet de représenter les valeurs booléennes True (vrai) et False (faux).
- ✓ Le type **float** est utilisé pour représenter les **nombres réels**. Attention, en Python, **le point** remplace la **virgule** utilisée en mathématiques (donc 5,8 est représenté par 5.8 en Python).
- ✓ Le type **str**, abréviation de string, est utilisé pour représenter des chaînes de caractères. On utilise des guillemets, des apostrophes ou str comme `a='bonjour'`, `a="bonjour"` ou `a=str(bonjour)`.

b. Les opérations sur les types numériques

Les opérations de base

+	addition
-	soustraction
*	multiplication
/	division
**	puissance
//	division entière
%	reste de la division

```
15 // 6
```

Réponse : 2

```
15 % 6
```

Réponse : 3

L'écriture scientifique

Exemple :

```
2.75e3
```

Réponse : 2750.0

- **Les opérateurs mathématiques classiques de comparaisons** `=`, `≠`, `<`, `≤`, `>`, `≥` s'écrivent en Python respectivement `==`, `!=`, `<`, `<=`, `>`, `>=`.
- **Les opérations logiques :**
 - **a and b** prend la valeur **True** si **a** et **b** sont **True** et sinon prend la valeur **False** ;
 - **a or b** prend la valeur **False** si **a** et **b** sont **False** et sinon prend la valeur **True** ;
 - **not a** prend la valeur **True** si **a** est **False** et prend la valeur **False** si **a** est **True**.

Remarques :

- Sur Python, pour utiliser des fonctions mathématiques comme la racine carré, il faut l'importer à partir du module math avec l'instruction suivante `from math import sqrt` (sqrt étant la racine carré en python)
- Pour importer toutes les fonctions du module math, la syntaxe est `from math import *` (à mettre au début du programme : 1^{ère} ligne en général)

Ce qu'il faut savoir : En général, un **algorithme** est construit en trois étapes :

- **Entrée** : On saisit les données.
- **Initialisation** : Le programme attribue des valeurs initiales à des variables. Ces valeurs peuvent changer au cours du programme.
- **Traitement des données** : Les instructions du programme effectuent des opérations à partir des données saisies dans le but de résoudre un problème.
- **Sortie** : Les résultats sont affichés

2. Affectation de variables

Définition :

Une **variable** est composée d'un nom (ou identificateur), d'une adresse en mémoire où est enregistrée une valeur ou un ensemble de valeurs) et d'un type qui définit ses propriétés.

Une **variable** permet de stocker une valeur que l'on compte réutiliser plus tard.

Les instructions de base sur les variables sont les suivantes :

- la **saisie** : on demande à l'utilisateur de donner une valeur à une variable ;
- l'**affectation** : est une instruction qui commande à la machine de créer une variable en lui précisant son nom et sa valeur) ;
- l'**affichage** : on affiche la valeur de la variable.

○

Remarques

Les notations ci-dessous sont équivalentes :

- Affecter à b la valeur de a
- b prend la valeur de a

En Python

- L'**affectation** se fait avec le signe `=` (**exemple :** `u=5` : on affecte ici la valeur 5 à la variable u)
- La **saisie** se fait avec la commande `input ()` : elle permet d'inviter l'utilisateur à saisir une valeur à l'exécution du programme.

Exemple :

```
nom = input("Quel est  
votre nom ?")
```

Remarque :

input est une fonction qui renvoie toujours une chaîne de caractères. Pour changer le type de la variable, on utilise : int (pour les entiers) ou float (pour les réels). Voir l'exemple ci-dessous :

```
n = float(input("Entrer un nombre"))  
print("Le carré de", n,"est", n * n)
```

- L'affichage se fait avec la commande **print ()**

```
print("Bonjour !")
```

```
print(2)
```

```
a = -3  
print("Le carré de", a,"est", a * a)
```

Remarque :

La commande **print ()** permet d'afficher textuellement la partie entre les guillemets " " et d'afficher la valeur des variables en dehors des guillemets. En effet dans le dernier exemple ci-dessus, il sera affiché : Le carré de -3 est 9.

Ex 3 - On donne l'algorithme suivant :

```
Lire  $n$   
 $q$  prend la valeur de  $(n + 2) \times (n + 2)$   
 $q$  prend la valeur de  $q - (n + 4)$   
 $q$  prend la valeur de  $q / (n + 3)$   
Afficher  $q$ 
```

- 1) Tester cet algorithme pour $n = 4$, puis pour $n = 7$.
- 2) Ecrire le programme python correspondant à cet algorithme (en considérant n comme un réel) et tester sur Python pour $n = 6,2$ puis pour $n = -9,8$
- 3) Émettre une conjecture sur le résultat fourni par cet algorithme puis démontrer cette conjecture.
- 4) Un élève a saisi $n = -3$. Que se passe-t-il ? Pourquoi ?

Ex 4 - Un commerçant accorde une remise sur des articles. On souhaite connaître le montant de la remise en euros. Voici un algorithme écrit en langage naturel donnant la solution au problème :

Entrée

Saisir le prix de départ A
Saisir le pourcentage de remise P

Traitement des données

Affecter au montant de la remise R la valeur $A \times \frac{P}{100}$

.....

Sortie

Afficher R

.....

- 1) Calculer la valeur de la variable R lorsque A = 56 et P = 30.
Donner une interprétation concrète du résultat précédent.
- 2) Compléter l'algorithme pour qu'il affiche également le prix à payer B.
- 3) Traduire l'algorithme en langage Python puis calculer la valeur des variables R et B lorsque A = 159 et P = 24. Donner une interprétation concrète des résultats précédents.
- 4) Même question avec A = 13 et P = 45

Exercice 5 :

- 1) Rédiger en langage naturel puis en langage Python un algorithme permettant de calculer le pourcentage de réduction d'un article connaissant le prix de départ et le prix à payer.

Entrée

.....
.....
.....

Traitement des données

.....
.....
.....

Sortie

.....
.....

- 2) Calculer avec le pourcentage de réduction d'un article dont le prix de départ est de 75 euros et le prix final est de 49,5 euros.

Définition : La résolution de certains problèmes nécessite la mise en place d'un **test** pour savoir si l'on doit effectuer une tâche.

Si la condition est remplie alors la tâche sera effectuée, sinon on effectue (éventuellement) une autre tâche. Dans un algorithme, on code la structure du « Si... AlorsSinon... FinSi » sous la forme suivante :

```
Si      condition Alors
        Tâche 1
        Tâche 2
Sinon
        Tâche 1bis
        Tâche 2bis
FinSi
```

Remarques :

- A chaque « Si » doit correspondre un « Fin Si ».
- En revanche le « Sinon » n'est pas toujours obligatoire. S'il n'est pas présent, aucune tâche ne sera effectuée si la condition n'est pas remplie.
- Il est important de respecter les espaces laissés au début de chaque ligne car ils permettent une meilleure lisibilité de l'algorithme.

En Python

- La commande **if** permet de tester le contenu d'une variable et exécute une série d'instructions si les conditions sont remplies.
- L'indentation, décalage vers la droite du début de ligne, est un élément de syntaxe important en Python . Elle délimite des blocs de code et elle aide à la lisibilité en permettant d'identifier facilement ces blocs. La ligne précédant l'indentation se termine par le signe deux-points .

La structure la plus simple est :

```
if condition:
    instructions
```

Le mot condition désigne une expression et le mot instructions désigne une instruction ou un bloc d'instructions écrites sur plusieurs lignes. Voici un exemple :

```
if n % 2 == 0:
    n = n // 2
```

La structure if-else présente une alternative :

```
if condition:
    instructions1
else:
    instructions2
```

Par exemple :

```
if n % 2 == 0:
    n = n // 2
else:
    n = 3 * n + 1
```

La structure if-elif-else présente plusieurs alternatives :

```
if condition1:
    instructions1
elif condition2:
    instructions2
elif condition3:
    instructions3
...
else:
    instructions
```

Par exemple :

```
if n % 4 == 0:
    n = n // 4
elif n % 4 == 1:
    n = (3 * n + 1) // 4
elif n % 4 == 2:
    n = n // 2
else:
    n = (3 * n + 1) // 2
```

Remarques :

- Les lignes qui commencent par les mots if et elif se terminent par le signe deux-points ;
- Le mot else est suivi immédiatement par le signe deux-points ;
- C'est l'indentation qui permet de délimiter les blocs d'instruction à exécuter si la condition est vérifiée.

Exercices d'application : test if

Exercice 1

Afficher « Entrez les dimensions du triangle en commençant par la plus grande »
Lire a , b et c
Si
 Afficher « Le triangle est rectangle. »
Sinon
 Afficher « Le triangle n'est pas rectangle. »
FinSi

- 1) Quelle condition faut-il écrire après le « Si » ?
- 2) Traduire cet algorithme en langage python puis tester pour $a=5$, $b=4$ et $c=3$
- 3) Modifier cet algorithme de façon à afficher en plus « Données incorrectes » si a n'est pas le plus grand des trois nombres rentrés.
- 4) Traduire ce nouvel algorithme en langage python puis tester pour $a=4$, $b=5$ et $c=3$.
Puis pour $a=7$, $b=4$, $c=6$ et enfin tester pour $a=8,5$; $b=3,6$ et $c=7,7$.

Exercice 2 :

Ex 5 - Un particulier souhaite louer une voiture.

- L'agence de location A demande 100 € au départ et 0,20 € par kilomètre parcouru ;
- L'agence de location B demande 150 € au départ et 0,15 € par kilomètre parcouru.

- 1) Compléter cet algorithme pour aider le consommateur à trouver la formule la plus avantageuse
- 2) Traduire cet algo en langage python puis donner la meilleure formule quand on veut parcourir 1000km, puis pour 525,75 km et enfin pour 1500km.

Afficher « Entrer le nombre de kilomètres. »
Lire n
Affecter à a la valeur $100 + 0,20 \times n$
Affecter à b la valeur
Si
 Afficher « La formule A est plus avantageuse. »
Sinon
 Si
 Afficher « Les formules A et B sont équivalentes. »
 Sinon
 Afficher « La formule B est plus avantageuse. »
FinSi
FinSi

Exercice 3

Un magasin propose de tirer des photos sur papier au tarif de 0,16 € la photo pour les 75 premières photos, puis 0,12 € la photo pour les photos suivantes. Écrire un algorithme qui demande à l'utilisateur d'entrer le nombre n de tirages photos commandés et calcule le montant à payer. Traduire l'algorithme en langage python puis tester avec des valeurs inférieures à 75 puis avec des valeurs supérieures à 75

Exercice 4

Écrire un algorithme qui demande deux nombres, puis affiche la différence entre le plus grand et le plus petit. Traduire en Python puis tester...

Définitions : Les boucles permettent de répéter plusieurs fois une partie de l'algorithme.

Il existe deux sortes de boucles :

1. Boucle « Pour » :

Lorsque **le nombre de répétitions n est connu à l'avance**, on utilise un compteur initialisé à 1 et qui augmente automatiquement de 1 à chaque répétition jusqu'à n . On parle de boucle itérative.

Pour Variable allant de Valeur début à Valeur fin

Tâche 1

Tâche 2

Fin Pour

En Python :

La syntaxe est la suivante :

```
for i in range(n):
    instructions
```

La variable i est appelé un compteur. Elle prend successivement les valeurs 0, 1, ..., $n-1$.

Exemple 1: Ecrivez un programme qui copierait 20 fois « je ne dois pas mâcher du chewing-gum »

```
for i in range(1,21):
    print("je ne dois pas mâcher du chewing-gum")
print("fini")
```

ou

```
for i in range(0,20):
    print("je ne dois pas mâcher du chewing-gum")
print("fini")
```

2. Boucle « Tant que » :

Lorsque **le nombre de répétitions n n'est pas connu à l'avance**, il peut dépendre d'une condition. Le traitement est répété tant que la condition est vraie. Lorsqu'elle est fausse, on sort de la boucle. On parle de boucle conditionnelle.

Tant que Condition est vraie

Tâche 1

Tache 2

Fin Tant que

En Python :

La syntaxe est la suivante :

```
while condition:
    instructions
```

```
i = 1
while i <= 5:
    print(i)
    i = i + 1
print('Fin !')
```

Exemple 1 :

Que fait ce programme ?

Exemple 2 :

Les deux programmes suivant sont équivalents :

```
a=0
while a<10:
    print("boucle Tant que")
    a=a+1
print("Fin du programme")
```

```
for loop in range(10):
    print("boucle for")
print("Fin du programme")
```

Que font ces programmes ?

Les fonctions en Python

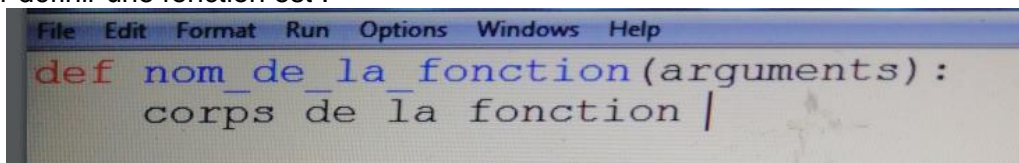
Ce qu'il faut savoir : Dans un programme, il est possible d'écrire des petits programmes ou des sous-programmes intermédiaires appelés *fonctions*.

Définition : Une fonction est un programme qui porte *un nom* et utilise zéro, une ou plusieurs variables appelés *paramètres ou arguments*.

Remarques :

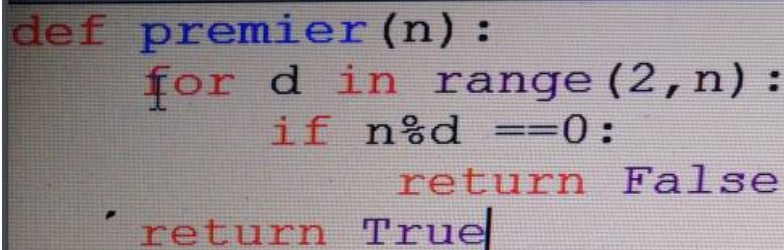
- Le nom d'une fonction ne peut pas avoir d'espace. On met donc des tirets « _ ».
- Si la fonction n'utilise aucun paramètre, on la note **def nom_de_la_fonction()**
- Les arguments (paramètres) sont séparés par des virgules. La première ligne, ligne de définition, se termine par le signe deux-points. Le corps de la fonction est un bloc indenté par rapport à la ligne de définition.
- L'instruction *return* permet de renvoyer une valeur de type entier, décimal (dit flottant) ou chaîne de caractères qui peut être réutilisée dans un autre programme ou une autre fonction. Elle interrompt le programme dès qu'elle s'est exécutée.

La syntaxe pour définir une fonction est :



```
File Edit Format Run Options Windows Help
def nom_de_la_fonction(arguments):
    corps de la fonction |
```

Exemple de fonction :



```
def premier(n):
    for d in range(2,n):
        if n%d ==0:
            return False
    return True
```

Que fait cette fonction ?

Exercices : Boucles « Pour »

Ex 1 - La somme des n premiers nombres entiers :

1	Lire n
2	Affecter à s la valeur 0
3	Pour i allant de 1 à n
4	Affecter à s la valeur $s + i$
5	Fin Pour
6	Afficher s

- 1) Tester cet algorithme pour $n = 3$ à l'aide d'un tableau où on donnera les valeurs de i et s après chaque exécution de la ligne 4.
- 2) Traduire cet algorithme en langage Python puis tester cet algorithme pour $n = 5$ puis pour $n=100$.

Ex 2 - On considère l'algorithme ci dessous :

1	Lire a et n
2	Affecter à p la valeur 1
3	Pour i allant de 1 à n
4	Affecter à p la valeur $p \times a$
5	Fin Pour
6	Afficher p

- 1) Tester cet algorithme pour $a = 2$ et $n = 3$ à l'aide d'un tableau où on donnera les valeurs de i et p après chaque exécution de la ligne 4.
- 2) Traduire cet algorithme en langage Python puis tester cet algorithme pour $a = 3$ et $n = 2$.
- 3) Que calcule cet algorithme ?

Ex 3 - Placement sur un compte épargne :

Fatou place 50 000 F CFA sur un compte épargne à 2% par an. Chaque année, les intérêts s'ajoutent au capital. Elle souhaite savoir quel sera son capital au bout de 7 ans.

1	Lire a et c
2	Pour i allant de 1 à
3	Affecter à c la valeur $c \times \dots$
4	Fin Pour
5	Afficher c

- 1) Compléter cet algorithme pour répondre au problème.
- 2) Tester cet algorithme pour $a = 7$ et $c = 50\,000$ à l'aide d'un tableau où on donnera les valeurs de a et c après chaque exécution de la ligne 3.
- 3) Programmer cet algorithme sur Python et contrôler la réponse au problème posé.

Ex 4 - Placement sur un compte épargne (suite) :

Reprendre l'exercice 3 :

- 1) Modifier l'algorithme (langage naturel et python) pour que l'utilisateur puisse choisir le taux de rémunération du compte épargne.
- 2) Tester l'algorithme pour un taux de 2,5% puis de 3%.

Ex 5 - Placement sur un compte épargne (suite) :

Reprendre l'exercice 4 : Fatou compte aussi placer 2 000 F CFA de plus par an.

- 1) Modifier l'algorithme (langage naturel et python) pour savoir quel sera alors son capital au bout de 7 ans.
- 2) Tester l'algorithme pour un taux de 2% ; 2,5% et 3%.

Ex 6 - Placement sur un compte épargne (suite) :

Reprendre l'exercice 5 :

- 1) Modifier l'algorithme (langage naturel et python) pour que l'utilisateur puisse entrer la somme qu'il veut placer en plus par an.
- 2) Tester l'algorithme pour un taux de 2,5% et un versement supplémentaire de 5 000 F CFA par an.

Exercices : Boucles « Tant que »

Ex 1 - Afficher les nombres entiers pairs inférieurs à un nombre choisi :

1	Lire n
2	Affecter à p la valeur 0
3	Tant que $n \geq p$
4	Afficher p
5	Affecter à p la valeur $p + 2$
6	Fin Tant que

- 1) Tester cet algorithme pour $n = 7$ à l'aide d'un tableau dans lequel on précisera les valeurs des différentes variables au niveau du « Fin Tant que ».
- 2) Programmer cet algorithme sur python et tester pour $n=18$, $n=47$...

Ex 2 - Placement sur un compte épargne :

Fatou place 50 000 F CFA sur un compte épargne à 5% par an. Chaque année, les intérêts s'ajoutent au capital. Elle souhaite savoir au bout de combien d'année son capital dépassera 100 000 F CFA et quel sera alors son capital.

1	Affecter à a la valeur 0
2	Lire c
3	Tant que $c < 100\ 000$
4	Affecter à c la valeur $c \times \dots$
5	Affecter à a la valeur $a + 1$
	Fin Tant que
	Afficher a
	Afficher c

- 1) Compléter cet algorithme pour répondre au problème.
- 2) Tester cet algorithme pour $c = 50\ 000$ à l'aide d'un tableau où on donnera les valeurs de a et c après chaque exécution de la ligne 3.
- 3) Programmer cet algorithme sur python et contrôler la réponse au problème posé.

Ex 3 - Placement sur un compte épargne (suite) :

Reprendre l'exercice 2 :

- 1) Modifier l'algorithme (langage naturel et python) pour que l'utilisateur puisse choisir le taux de rémunération du compte épargne.
- 2) Tester l'algorithme pour un taux de 2% puis de 3%.

Ex 4 - Placement sur un compte épargne (suite) :

Reprendre l'exercice 3 : Fatou compte aussi placer 2 000 F CFA de plus par an.

- 1) Modifier l'algorithme (langage naturel et python) pour savoir au bout de combien d'année son capital dépassera 100 000 F CFA et quel sera alors son capital.
- 2) Tester l'algorithme pour un taux de 2% ; 3% ; 5%.

Ex 5 - Placement sur un compte épargne (suite) :

Reprendre l'exercice 4 :

- 1) Modifier l'algorithme (langage naturel et python) pour que l'utilisateur puisse entrer la somme qu'il veut placer en plus par an.
- 2) Tester l'algorithme pour un taux de 2,5% et un versement supplémentaire de 5 000 F CFA par an.

Exercices : Fonctions

Exercice 1 :

Soit le programme python suivant :

```
def carre(n):  
    return n*n
```

- Quel est le nom de la fonction définie dans le programme ci-dessus
- Cette fonction a-t-elle des paramètres ? Si oui combien ? et lesquels ?
- Que fait cette fonction ?
- Taper le programme sur Python et donner le résultat obtenu pour $n=8$ puis pour $n=6,2$

Exercice 2 :

Soient les deux programmes python suivants :

```
def premiers_carres(k):  
    for i in range(k):  
        print carre(i)
```

```
def premiers_carres(k):  
    i = 0  
    while i < k:  
        print carre(i)  
        i = i + 1
```

- Dans chacun de ces deux programmes est définie une fonction . Que fait la fonction du 1^{er} programme ? la fonction du 2^e programme ? Que remarquez-vous ?
- Taper chacun de ces deux programmes sur Python et donner le résultat obtenu pour chacun d'eux pour $n=5$ puis pour $n=12$
- Que faut-il rajouter au programme si on veut qu'il demande d'abord un entier k à l'utilisateur puis qu'il affiche par la suite les k premiers carrés

Exercice 3 :

Ecrire une fonction qui prend en paramètres deux nombres et renvoie le plus grand des deux nombres.

Exercice 4 :

Ecrire une fonction qui prend en paramètre un entier strictement positif k et qui renvoie la somme des k premiers carrés non nuls.