

Ministère de l'Éducation  
Direction Générale des Programmes  
et de la Formation Continue



# **IMPLEMENTATION EN PYTHON DES CONVENTIONS ALGORITHMIQUES**

**Année scolaire 2022/2023**

Le langage de programmation choisi pour implémenter les solutions algorithmiques est le langage de programmation Python.

## A. Introduction générale

- Python est un langage de programmation sensible à la casse.
- Dans un code Python, il est recommandé d'ajouter des commentaires.
  - Commentaire sur une seule ligne : débiter la ligne par le symbole #.
  - Commentaire sur plusieurs lignes : délimiter les lignes du commentaire par ""

## B. Les syntaxes des structures algorithmiques

### 1. Les opérations élémentaires simples

#### a. L'opération d'entrée

En algorithmique	En python
Lire (Objet)	Objet = <b>input()</b>  Objet = <b>input('message')</b>  <i>N.B. :</i> Par défaut, la valeur saisie est de type chaîne de caractères.

#### b. L'opération de sortie

En algorithmique	En python
Écrire ("Message", Objet, Expression)	<b>print</b> ("Message", Objet, Expression)
Écrire_nl ("Message", Objet, Expression)	<b>print</b> ("Message", Objet, Expression, "\n")

#### Remarques :

- Objet est de type variable simple (entier, réel, booléen, caractère et chaîne de caractères).
- "\n" permet d'ajouter un retour à la ligne.
- L'affichage d'un tableau T en python, doit se faire élément par élément et non pas avec l'instruction **print(T)**.

#### c. L'opération d'affectation

En algorithmique	En python
Objet ← Expression	Objet = Expression

**Remarque :** **Objet** est une variable de type simple (entier, réel, booléen, caractère et chaîne de caractères).

## 2. Les types de données simples

En algorithmique	En python
Entier	<b>int</b>
Réel	<b>float</b>
Booléen	<b>bool</b>
Caractère	<b>str</b>
Chaîne de caractères	<b>str</b>

### Exemples de conversions entre les types simples en python

Conversion	Syntaxe	Exemple
De <b>str</b> vers <b>int</b>	int(ch)	x = int("3") signifie que x reçoit l'entier 3
De <b>str</b> vers <b>float</b>	float(ch)	x = float("3.2") signifie que x reçoit le réel 3.2
De <b>str</b> vers <b>bool</b>	bool(ch)	x = bool("0") signifie que x reçoit True
De <b>int</b> vers <b>str</b>	str(int)	x = str(3) signifie que x reçoit le caractère "3"
		x = str(123) signifie que x reçoit la chaîne "123"

## 3. Les structures de données

En algorithmique	En python
Tableau (à une et à deux dimensions)	Ces types seront présentés ci-après ( <b>voir 4.b) , 4.c) et 4.d).</b>
Enregistrement	
Fichier	

## 4. Les déclarations

### a. Les objets de type de donnée simple

En algorithmique					
	<table><tr><th>Objet</th><th>Type / Nature</th></tr><tr><td>Nom_objet</td><td>Type_objet</td></tr></table>	Objet	Type / Nature	Nom_objet	Type_objet
Objet	Type / Nature				
Nom_objet	Type_objet				
En Python					
Une variable <b>n'a pas besoin d'être déclarée</b> avec un type particulier : c'est au moment où on lui attribue une valeur qu'elle sera créée. Ainsi, son type sera défini en fonction du type de la valeur qui lui a été attribuée. L'identificateur d'une variable est <b>sensible à la casse</b> .					

## b. Les tableaux

En algorithmique		
	Objet	Type / Nature
Tableau à une dimension	Nom_tableau	Tableau de N Type _élément
Tableau à deux dimensions	Nom_tableau	Tableau de N lignes * M colonnes Type _élément

En Python			
<ul style="list-style-type: none"><li>On utilisera la bibliothèque <b>numpy</b> pour implémenter les tableaux.</li><li>Un tableau de la bibliothèque numpy est :<ul style="list-style-type: none"><li><b>homogène</b>, c'est-à-dire constitué d'éléments de <b>même type</b>,</li><li><b>statique</b>, car sa taille est <b>fixée lors de la création</b>.</li></ul></li><li>La déclaration d'un tableau se fait en deux étapes :<ul style="list-style-type: none"><li><u><b>Importation</b></u> des modules nécessaires de la bibliothèque <b>numpy</b></li></ul></li></ul>			
	<table><tr><th>Importation</th></tr><tr><td><pre>from numpy import array ou from numpy import * ou import numpy as alias</pre></td></tr></table>	Importation	<pre>from numpy import array ou from numpy import * ou import numpy as alias</pre>
Importation			
<pre>from numpy import array ou from numpy import * ou import numpy as alias</pre>			
	<ul style="list-style-type: none"><li><u><b>Déclaration</b></u> du tableau</li></ul>		
Tableau à une dimension	<table><tr><th>Déclaration</th></tr><tr><td><pre>T = array ([Type_élément] * N) ou bien T = array ([valeur_initiale] * N)</pre></td></tr></table>	Déclaration	<pre>T = array ([Type_élément] * N) ou bien T = array ([valeur_initiale] * N)</pre>
Déclaration			
<pre>T = array ([Type_élément] * N) ou bien T = array ([valeur_initiale] * N)</pre>			
Tableau à deux dimensions	<table><tr><td><pre>T = array ([Type_élément]*Colonnes]* Lignes) ou bien T = array ([valeur_initiale]*Colonnes]* Lignes)</pre></td></tr></table>	<pre>T = array ([Type_élément]*Colonnes]* Lignes) ou bien T = array ([valeur_initiale]*Colonnes]* Lignes)</pre>	
<pre>T = array ([Type_élément]*Colonnes]* Lignes) ou bien T = array ([valeur_initiale]*Colonnes]* Lignes)</pre>			

**Remarque :** On peut spécifier le type des éléments d'un tableau avec la syntaxe :

*Nom\_tableau = array ([Valeur\_initiale] \* N, dtype=Type\_élément)*

## Exemples de déclarations de tableaux en Python

Déclaration	Explication
<code>T = array ([5] * 10)</code>	Déclarer un tableau T de 10 entiers et initialiser ses éléments par « 5 ».
<code>T = array ([float ()] * 10)</code>	Déclarer un tableau T de 10 réels et initialiser ses éléments par «0.0 ».
<code>T = array ([str] * 10)</code>	Déclarer un tableau T de 10 chaînes de caractères.
<code>T = array ([str()] * 10)</code>	Déclarer un tableau T de 10 caractères et initialiser ses éléments par le caractère vide.
<code>T = array ([''] * 10 , dtype = 'U20')</code>	Déclarer un tableau T de 10 éléments initialisés par une chaîne vide. Chaque élément peut contenir 20 caractères au maximum.
<code>T = array ([[int ( )] * 10]*30)</code>	Déclarer un tableau T de 30 lignes x 10 colonnes d'entiers.

### c. L'enregistrement

En algorithmique					
<table> <tr> <th>Objet</th><th>Type / Nature</th></tr> <tr> <td>Nom_enregistrement</td><td>                     Enregistrement                          Nom_champ1 : Type_champ1                          Nom_champ2 : Type_champ2                          ...                          Fin                 </td></tr> </table>	Objet	Type / Nature	Nom_enregistrement	Enregistrement Nom_champ1 : Type_champ1 Nom_champ2 : Type_champ2 ... Fin	
Objet	Type / Nature				
Nom_enregistrement	Enregistrement Nom_champ1 : Type_champ1 Nom_champ2 : Type_champ2 ... Fin				
En Python					
<pre> Nom_enregistrement = dict (     Nom_champ1 = Type_champ1,     Nom_champ2 = Type_champ2,     ... )</pre>					

**Remarque :** Pour accéder à un champ d'un enregistrement on utilise la syntaxe suivante : `Nom_Enregistrement [ 'Nom_Champ' ]`.

#### d. Les fichiers

En algorithmique		
	Objet	Type / Nature
	Fichier texte	Nom_fichier
	Fichier de données	Fichier de Type _élément
En Python		
La déclaration d'un <b>objet de type fichier</b> se fait lors de sa création à l'aide de la <b>fonction open()</b> détaillée ci-après (voir 10.a) et 10.b) ).		

### 5. Les structures de contrôle conditionnelles

En algorithmique	En python
<b>Si</b> Condition <b>Alors</b> Traitement <b>FinSi</b>	<b>if</b> Condition : Traitement
<b>Si</b> Condition <b>Alors</b> Traitement1 <b>Sinon</b> Traitement2 <b>FinSi</b>	<b>if</b> Condition : Traitement1 <b>else :</b> Traitement2
<b>Si</b> Condition1 <b>Alors</b> Traitement1 <b>Sinon Si</b> Condition2 <b>Alors</b> Traitement2 ..... <b>[Sinon</b> TraitementN] <b>FinSi</b>	<b>if</b> Condition1 : Traitement1 <b>elif</b> Condition2 : Traitement2 ..... <b>else :</b> TraitementN
<b>Selon</b> <Sélecteur> Valeur1_1[, Valeur1_2, ...] : Traitement1 Valeur2_1 .. Valeur2_2 : Traitement2 ..... <b>[Sinon</b> TraitementN] <b>Fin Selon</b>	<b>A partir de la version 3.10</b> <b>match</b> Sélecteur : <b>case</b> Valeur1 : Traitement1 <b>case</b> Valeur2_1   Valeur2_2 : Traitement2 <b>case</b> Sélecteur if V3_1 <=Sélecteur<= V3_2 : Traitement3 <b>case</b> _ : TraitementN <b>N.B. : Le sélecteur doit être de type scalaire.</b>

## 6. Les structures de contrôle itératives

### a. La structure de contrôle itérative complète

En algorithmique
<b>Pour</b> compteur de Début à Fin [Pas = valeur_pas] Faire Traitement <b>Fin Pour</b>
En Python
<b>for</b> compteur <b>in range</b> (Début, Fin+1, Pas) : <b>Traitement</b> <i>N.B. : La valeur finale du compteur est exclue de la boucle.</i>

#### Remarques :

- La valeur du **pas** peut être **positive ou négative**. Par défaut, elle est égale à 1.
- Ne pas utiliser l'instruction **break** pour forcer l'arrêt de la boucle **for**.

### b. Les structures de contrôle itératives à condition d'arrêt

En algorithmique	En Python
<b>Tant que</b> Condition <b>Faire</b> Traitement <b>Fin Tant que</b>	<b>while</b> Condition :  <b>Traitement</b>
<b>Répéter</b> Traitement <b>Jusqu'à</b> Condition d'arrêt	

**Remarque :** Ne pas utiliser l'instruction **break** pour forcer l'arrêt de la boucle **while**.



## 7. Les modules

### a. La déclaration

En algorithmique	En Python
<b>Fonction</b> Nom_fonction (pf <sub>1</sub> : type <sub>1</sub> , pf <sub>2</sub> : type <sub>2</sub> , ... , pf <sub>n</sub> : type <sub>n</sub> ) : <b>Type_résultat</b> <b>DEBUT</b> Traitement <b>Retourner</b> résultat <b>FIN</b>	Un module (fonction ou procédure) se définit en utilisant le mot clé <b>def</b> selon la syntaxe suivante :  <b>def</b> Nom_module (pf <sub>1</sub> , pf <sub>2</sub> , ... , pf <sub>n</sub> ) : <b>Traitement</b> <b>[return résultat]</b>
<b>Procédure</b> Nom_procédure (pf <sub>1</sub> : type <sub>1</sub> , pf <sub>2</sub> : type <sub>2</sub> , ... , pf <sub>n</sub> : type <sub>n</sub> ) <b>DEBUT</b> Traitement <b>FIN</b>	<b>N.B. :</b> Dans un module, l'instruction " <b>return</b> " peut être utilisée, et ce, pour retourner <b>un seul résultat de type simple</b> .

### b. L'appel

Module	En algorithmique	En Python
Fonction	Objet <b>←</b> Nom_fonction (pe <sub>1</sub> , ..., pe <sub>n</sub> )	<b>Objet =</b> Nom_module (pe <sub>1</sub> , ..., pe <sub>n</sub> )
Procédure	Nom_procédure (pe <sub>1</sub> , ... , pe <sub>n</sub> )	<b>Nom_module</b> (pe <sub>1</sub> , ... , pe <sub>n</sub> )

### c. Le mode de passage

En algorithmique	En Python
Si le mode de passage est par référence (par adresse), on ajoutera le symbole @ avant le nom du paramètre.  <b>Procédure</b> Nom_procédure (@pf <sub>1</sub> : type <sub>1</sub> , @pf <sub>2</sub> : type <sub>2</sub> , ... , pf <sub>n</sub> : type <sub>n</sub> ) <b>DEBUT</b> Traitement <b>FIN</b>	Nom_module (pf <sub>1</sub> , pf <sub>2</sub> , ... , pf <sub>n</sub> ) : Traitement  <b>N.B. :</b> En python, les paramètres de type <b>dictionnaire</b> , <b>tableau</b> et <b>fichier</b> sont, par défaut passés par <b>référence</b> .

### d. La portée des variables en python :

- Toute variable déclarée au sein d'un module a une **portée locale**.
- Toute variable déclarée au sein d'un module précédée par le mot clé **global** a une **portée globale**. Par conséquent, elle ne devra pas figurer parmi les paramètres de ce module.



## Exemple d'un programme en python présentant une solution modulaire

```
from numpy import array
T1 = array([0]*10) #Déclaration du tableau T1
T2 = array([0]*15) #Déclaration du tableau T2
#définition du module saisieTaille
def saisieTaille (bornInf , bornSup) :
    taille = 0
    while taille not in range(bornInf, bornSup+1):
        taille=int(input("Taille entre "+str(bornInf)+" et "+str(bornSup)+" :"))
    return taille
#définition du module remplirTab
def remplirTab(T, taille) :
    for i in range( taille ) :
        T[i] = int( input ("Donner l'élément N° " + str(i) + " : "))
#définition du module afficherTab
def afficherTab(T , taille) :
    for i in range(taille) :
        print(T[i])
#Le programme principal
n = saisieTaille (5,10) #1er appel du module saisieTaille
m = saisieTaille (3,15) #2ème appel du module saisieTaille
print("chargement de T1")
remplirTab(T1 , n)
print("chargement de T2")
remplirTab(T2 , m)
print("Affichage du tableau T1")
afficherTab(T1 , n)
print("Affichage du tableau T2")
afficherTab(T2 , m)
```

2ème façon d'implémentation du module saisieTaille en utilisant une variable globale nommée Taille

```
#définition du module saisieTaille
def saisieTaille (bornInf , bornSup) :
    global taille
    taille = 0
    while taille not in range( bornInf , bornSup+1 ) :
        taille=int(input("Taille entre "+str(bornInf)+" et "+str(bornSup)+" :"))
#Le programme principal
saisieTaille (5 , 10) #1er appel du module saisieTaille
n = taille
saisieTaille (3 , 15) #2ème appel du module saisieTaille
m = taille
print("chargement de T1")
remplirTab(T1 , n)
print("chargement de T2")
remplirTab(T2 , m)
print("Affichage du tableau T1")
afficherTab(T1 , n)
print("Affichage du tableau T2")
afficherTab(T2 , m)
```

## 8. Les opérateurs arithmétiques et logiques

### a. Opérateurs arithmétiques

Opération	En algorithmique	En Python
Somme	+	+
Soustraction	-	-
Multiplication	*	*
Division	/	/
Division entière	Div	//
Reste de la division entière	Mod	%

### b. Opérateurs de comparaison

Opération	En algorithmique	En Python
Egal	=	==
Différent	≠	!=
Strictement supérieur	>	>
Supérieur ou égal	≥	>=
Strictement inférieur	<	<
Inférieur ou égal	≤	<=
Appartient ( <b>entier, caractère</b> )	∈	in

### c. Opérateurs logiques

Opération	En algorithmique	En Python
Négation	Non	not
Conjonction	Et	and
Disjonction	Ou	or

## 9. Les fonctions prédéfinies

### a. Les fonctions sur le type numérique

En algorithmique	En Python	Observation
Arrondi (x)	round (x)	
RacineCarré(x)	sqrt (x)	Nécessite l'importation de la bibliothèque <b>math</b> .
Aléa (vi, vf)	randint(vi, vf)	Nécessite l'importation de la bibliothèque <b>random</b> .
Ent(x)	int (x)	
Abs (x)	abs (x)	

### b. Les fonctions sur le type caractère

En algorithmique	En Python
<b>Ord</b> (c)	<b>ord</b> (c)
<b>Chr</b> (d)	<b>chr</b> (d)

### c. Les fonctions sur le type chaîne de caractères

En algorithmique	En Python
<b>Long</b> (ch)	<b>len</b> (ch)
<b>Pos</b> (ch1, ch2)	<b>ch2.find</b> (ch1)
<b>Convch</b> (x)	<b>str</b> (x)
<b>Estnum</b> (ch)	<b>ch.isdecimal</b> ()
<b>Valeur</b> (ch)	<b>int</b> (ch)   <b>float</b> (ch)
<b>Sous_chaine</b> (ch, d, f)	<b>ch[d:f]</b>
<b>Effacer</b> (ch, d, f)	<b>ch = ch[ : d ]+ch[ f :]</b>
<b>Majus</b> (ch)	<b>ch.upper</b> ()

**Remarque :** Pour concaténer deux chaînes de caractères, on utilise l'opérateur « + ».

## 10. Les fonctions et les procédures prédéfinies sur les fichiers

### a. Les fichiers de données

En algorithmique	En Python
<b>Ouvrir</b> ("Chemin\Nom_physique", Nom_logique , "Mode") Avec mode d'ouverture égal à : <ul style="list-style-type: none"> <li>○ "rb" : Lecture (pointer au début)</li> <li>○ "wb" : Ecriture (création)</li> <li>○ "ab" : Ajout à la fin du fichier</li> </ul>	<b>Nom_logique=open</b> ('Chemin\Nom_physique' , 'Mode')
<b>Lire</b> (Nom_logique , Objet)	from pickle import <b>load, dump</b> Objet = <b>load</b> (Nom_logique)
<b>Ecrire</b> (Nom_logique , Objet)	from pickle import <b>load, dump</b> <b>dump</b> (Objet , Nom_logique)
<b>Fin_fichier</b> (Nom_logique)	Fin_fichier = False <b>while</b> not (Fin_fichier) : try : x = <b>load</b> (Nom_logique) except : Fin_fichier = True
<b>Fermer</b> (Nom_logique)	<b>Nom_logique.close</b> ()

## b. Les fichiers textes

En algorithmique	En Python
<b>Ouvrir</b> ("Chemin\Nom_physique" , Nom_logique , "Mode") Avec mode d'ouverture égal à : <ul style="list-style-type: none"><li>○ "r" : Lecture</li><li>○ "w" : Ecriture (création)</li><li>○ "a" : Ajout à la fin du fichier</li></ul>	Nom_logique = <b>open</b> ( 'Chemin\Nom_physique' , 'Mode' )
<b>Lire</b> (Nom_logique , ch)	ch = Nom_logique. <b>read()</b>
<b>Lire_ligne</b> (Nom_logique , ch)	ch = Nom_logique. <b>readline()</b>
<b>Ecrire</b> (Nom_logique , ch)	Nom_logique. <b>write</b> (ch)
<b>Ecrire_nl</b> (Nom_logique , ch)	Nom_logique. <b>write</b> (ch + "\n")
<b>Fin_fichier</b> (Nom_logique)	ch= Nom_logique. <b>readline()</b> <b>While</b> ch != "" : Traitement ch = Nom_logique. <b>readline()</b> <b>N.B. : La fin d'un fichier texte est la chaîne vide</b>
<b>Fermer</b> (Nom_logique)	Nom_logique. <b>close ()</b>

**Remarque :** Lors de la résolution d'un problème, il est fortement **interdit d'utiliser autres fonctions ne figurant pas dans la liste des fonctions énumérées** dans le présent document. Toutefois, les énoncés des épreuves pratiques du baccalauréat des matières « Informatique » et « Algorithmique et programmation », pourraient intégrer une **nouvelle fonction**. Dans ce cas, le **rôle** et la **syntaxe** de cette fonction seront détaillés dans l'énoncé de l'épreuve.