Dans le but de développer le raisonnement et la capacité de résolution des problèmes chez l'apprenant, le domaine « **Pensée computationnelle et programmation** » met l'accent sur l'algorithmique. L'écriture de l'algorithme **doit respecter** les conventions citées dans ce document.

1. La forme générale d'un algorithme

ALGORITHME *Nom*
DEBUT
Traitement1
TraitementN
FIN

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**

N.B. :

- L'écriture de l'algorithme doit obligatoirement respecter **l'indentation**.
- Le nommage des objets doit être significatif.

2. Les opérations élémentaires simples

L'opération d'entrée

Notation Algorithmique	Notation en Python
Ecrire ("Commentaire") **Lire** (Objet)	*Pour les chaînes de caractères :* Objet = **input** ("Commentaire") *Pour les entiers :* Objet = **int** (**input** ("Commentaire")) *Pour les réels :* Objet = **float** (**input** ("Commentaire"))

L'opération de sortie

Notation Algorithmique	Notation en Python
Ecrire ("Message", Objet) **Ecrire** ("Message", Expression)	**print** ("Message", Objet, **end** = "") **print** ("Message", Expression, **end** = "")
Ecrire_nl ("Message", Objet) **Ecrire_nl** ("Message", Expression)	**print** ("Message", Objet) **print** ("Message", Expression) ***N.B. :*** *"print" fait un retour à la ligne automatique*

N.B. : Objet est de type simple.

L'opération d'affectation

Notation Algorithmique	Notation en Python
Objet ← Valeur	Objet = Valeur
Objet ← Expression	Objet = Expression
Objet1 ← Objet2	Objet1 = Objet2

N.B. : Objet1 et Objet2 doivent être de même type ou de types compatibles.

3. Les types de données simples

3.1. Les types des données

Types des données en algorithmique	Types des données en Python
Entier	**int** ( )
Réel	**float** ( )
Caractère	**str** ( )
Booléen	**bool** ( )
Chaîne de caractères	**str** ( )

3.2. Les déclarations des objets en algorithmique

La déclaration des **constantes** et des **variables** est réalisée comme suit :

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Constante	Constante = Valeur de la Constante
Nom_Variable	Type_Variable

N.B. :

- L'indice du 1er caractère d'une chaîne de caractère est **Zéro**.
- Pour accéder à un caractère d'une chaîne **Ch**, on utilise la notation **Ch[i]** avec $\mathbf{0 \leq i \leq long(Ch)-1}$.
- On pourra utiliser l'opérateur « + » pour concaténer deux chaînes.

4. Les opérateurs arithmétiques et logiques

- Toute structure algorithmique à laquelle il n'y a pas de correspondance au niveau du langage de programmation utilisé sera remplacée par la structure adéquate de ce dernier.
- Pour les opérateurs arithmétiques et logiques, on va se baser sur celles qui sont standards et développés dans le tableau suivant :

Les opérateurs arithmétiques et logiques et leurs priorités				
Désignation de l'opération	**Priorité**	**Notation**		**Type d'opérande**
		Algo.	**Python**	
Parenthèses	**1**	(…)	(…)	Tous les types
Multiplication	**2**	*	*	Entier ou Réel
Division réelle		/	/	Réel
Division entière		**Div**	//	Entier
Reste de la division entière		**Mod**	%	Entier
Addition	**3**	+	+	Entier ou Réel
Soustraction		-	-	Entier ou Réel
Égale	**4**	=	==	Tout Type ordonné
Différent		≠	!=	Tout Type ordonné
Strictement supérieur		>	>	Tout Type ordonné
Supérieur ou égal		≥	>=	Tout Type ordonné
Strictement inférieur		<	<	Tout Type ordonné
Inférieur ou égal		≤	<=	Tout Type ordonné
L'appartenance (Entier, Caractère ou Booléen)		**∈**	**in**	Type Scalaire

Remarques :

Appartenance	Notation	
	Algorithmique	**Python**
Ensemble	x ∈ {val1, val2, …, valn}	**x in {val1, val2, …}**
Intervalle	x ∈ [val1..valn] *ou bien* **val1 ≤ x ≤ valn**	Pour les entiers : **x in range(val1, valn+1)** *ou bien* **val1 <= x <= valn** Pour les caractères : **ord(x) in range(ord(val1),ord(valn)+1)** *ou bien* **val1 <= x <= valn**

Opérateurs logiques (booléens) & Priorités & Tables de vérité						
Opération	**Opérateurs & Priorités & Tables de vérité**					
	Algo.	**Python**	**Priorité**	**Table de vérité**		
Négation	**NON**	**not**	**1**	**A**	**not (A)**	
				True	**False**	
				False	**True**	
Conjonction	**ET**	**and**	**2**	**A**	**B**	**A and B**
				True	True	**True**
				True	False	**False**
				False	True	**False**
				False	False	**False**
Disjonction	**OU**	**or**	**3**	**A**	**B**	**A or B**
				True	True	**True**
				True	False	**True**
				False	True	**True**
				False	False	**False**

5. Les fonctions prédéfinies

5.1. Les fonctions arithmétiques

Les fonctions sur les types numériques				
Notation algorithmique	**Notation Python**	**Rôle**	**Exemples en Python - Résultat**	
N ← **abs** (X)	N = **abs** (X)	Retourne la valeur absolue de **X**.	N = abs (-20) N = abs (5.8)	N == 20 N == 5.8
N ← **ent** (X)	N = **int** (X)	Retourne un **Entier** représentant la partie entière de **X**.	N = int (5.2) N = int (-5.8)	N == 5 N == -5
N ← **arrondi** (X)	N = **round** (X)	Retourne **l'Entier** le plus proche de **X**. **N.B.** : En Python, si la partie fractionnaire est égale à **5**, l'entier **Pair** le plus proche est retourné.	N = round (2.2) N = round (2.8) N = round (2.5) N = round (3.5)	N == 2 N == 3 N == 2 N == 4
N ← **racinecarré** (X)	**from math import sqrt** N = **sqrt** (X)	Retourne un **Réel** représentant la racine carrée de **X**. Si **X** < 0, elle provoque une erreur.	N = sqrt (9) N = sqrt (25.0) N = sqrt (-5)	N == 3.0 N == 5.0 Erreur
N ← **aléa** (Vi , Vf)	**from random import randint** N = **randint** (Vi , Vf)	Retourne un entier d'une façon aléatoire et automatique de l'intervalle [**Vi** , **Vf**].	N = randint (2 , 5) N pourra avoir 2 ou 3 ou 4 ou 5	

5.2. Les fonctions sur les caractères

Les fonctions sur le type caractère				
Notation algorithmique	**Notation Python**	**Rôle**	**Exemples en Python - Résultat**	
N ← **Ord** (Ca)	N = **ord** (Ca)	Retourne le code ASCII du caractère **Ca**.	N = ord ("0") N = ord ("A") N = ord ("a")	N == 48 N == 65 N == 97
Ca ← **Chr** (X)	Ca = **chr** (X)	Retourne le Caractère dont le code ASCII est **X**.	Ca = chr (50) Ca = chr (90)	Ca == "2" Ca == "Z"

5.3. Les fonctions sur les chaînes de caractères

Les fonctions sur le type chaîne de caractères				
Notation algorithmique	**Notation Python**	**Rôle**	**Exemples en Python - Résultat**	
Lo ← **long** (Ch)	Lo = **len** (Ch)	Retourne un entier représentant le nombre de caractères de la chaîne **Ch** (la longueur de Ch).	Lo = len ("Salut") Lo = len ("L'élève") Lo = len ("")	Lo == 5 Lo == 7 Lo == 0
Po ← **pos** (Ch1, Ch2)	Po = Ch2.**find** (Ch1)	Retourne un entier représentant la position de la 1ère occurrence de **Ch1** dans **Ch2**. Elle retourne **-1** si **Ch1** n'existe pas dans **Ch2**.	Ch1 = "Y" Ch2 = "BAYBAY" Po = Ch2.find (Ch1)	Po == 2
Ch2 ← **sous_chaîne** (Ch1, Début, Fin)	Ch2 = Ch1 [Début : Fin]	Retourne une copie de la chaîne **Ch1** à partir de l'indice **Début** à l'indice **Fin** (position **Fin** exclu)	Ch1 = "BACCALAUREAT" Ch2 = Ch1 [5 : 12]	Chr == "LAUREAT"
Ch2 ← **effacer** (Ch1, d, f)	Ch2= Ch1[: d]+Ch1 [f:]	Retourne une chaîne **Ch2** après avoir effacer, de la chaîne **Ch1**, les caractères de la position **d** à la position **f** (**f** exclu).	Ch1 = "INFORMATIQUE" Ch2 = Ch1[:6] + Ch1[11:]	Ch2 == "INFORME"
ChM ← **majus** (Ch)	ChM = Ch.**upper** ( )	Retourne la chaîne **ChM** représentant la conversion en Majuscule de la chaîne **Ch**.	Ch = "Bonjour" ChM = Ch.upper ( )	ChM =="BONJOUR"
Ch ← **convch** (**X**)	Ch = **str** (**X**)	Retourne la conversion du nombre **X** en une chaîne de caractères.	N = 358 Ch = str (N)	Ch == "358"
Test ← **estnum** (Ch)	**Pas de correspondance**. Toutefois, on pourra utiliser **isnumeric** ( ) malgré qu'elle ne répond pas aux exigences ou bien développer un module qui permet de réaliser cette tâche.	Retourne **VRAI** si la chaîne **Ch** est convertible en une valeur numérique et **FAUX** dans le cas contraire.	Ch = "489" Test = Ch . isnumeric ( ) Ch = "489.56" Test = Ch . isnumeric ( )	Test == True Test == False
N ← **valeur** (Ch)	N = **int** (Ch) *ou bien* N = **float** (Ch)	Retourne la conversion d'une chaîne **Ch** en une valeur numérique, si c'est possible.	Ch = "489" N = int (Ch) Ch = "489" N = float (Ch)	N == 489 N == 489.0

6. Les types de données structurées et leurs déclarations

6.1. Les tableaux à une seule dimension

6.1.1. Déclaration en algorithmique

❖ 1ère méthode

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Tableau	**Tableau** de **N Type_élément**

❖ 2ème méthode

Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Nom_Type_Tableau = **Tableau** de **N Type_élément**

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Tableau	Nom_Type_Tableau

6.1.2. Déclaration en Python en utilisant la bibliothèque Numpy

Déclaration dans le cas général
import numpy as np **Nom_Tableau = np.array ([Type_élément ( )] * N [,dtype = object])**

Exemples de déclarations en Python	
Déclaration	**Explication**
import numpy as np T = np.array ([int ( )] * 20)	Pour déclarer un tableau de 20 entiers avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([float ()] * 100)	Pour déclarer un tableau de 100 réels avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([str ( )] * 50 , dtype = object) *ou bien* T = np.array ([str] * 50)	Pour déclarer un tableau de 50 chaînes de caractères avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([bool ( )] * 10)	Pour déclarer un tableau de 10 booléens avec importation de la bibliothèque Numpy.

N.B. :

- Pour initialiser un tableau, on pourra utiliser la déclaration suivante :

  Nom_Tableau = np.array ([valeur_initiale] * N , Type_élément)
- Les éléments d'un tableau à une seule dimension doivent être de **même type**.
- Les indices des éléments d'un tableau sont de **type scalaire**.
- En Python, pour déclarer un tableau de chaînes de caractères et éviter que chaque case contiendra uniquement le premier caractère de la chaîne saisie, on peut procéder comme suit :
  - ajouter "**dtype = object**" à la déclaration,
  - supprimer les parenthèses après le "**str**".
- L'indice de la 1ère case d'un tableau est, par défaut, égal à **Zéro**.
- Pour accéder à un élément d'indice « **i** » d'un tableau : **Nom_Tableau [i]**

6.2. Les tableaux à deux dimensions

6.2.1. Déclaration en algorithmique

- 1ère méthode

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Matrice	**Tableau** de **N** Lignes * **M** Colonnes **Type_élément**

- 2ème méthode

Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Nom_Type_Matrice = **Tableau** de **N** Lignes * **M** Colonnes **Type_élément**

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Matrice	Nom_Type_Matrice

6.2.2. Déclaration en Python en utilisant la bibliothèque Numpy

Déclaration dans le cas général
import numpy as np **Nom_Matrice = np.array ([[Type_élément ( )] * M] * N [,dtype = object]))** **N.B. :** • N est le nombre de lignes. • M est le nombre de colonnes. • Il est obligatoire d'importer la bibliothèque Numpy de Python.

Exemple de déclaration en Python
Pour déclarer une matrice M de 6 lignes et de 5 colonnes d'Entiers : **import** numpy **as** np M = np.**array** ([[int ( )] * 5] * 6)

N.B. :

- Pour initialiser une matrice, on pourra utiliser la déclaration suivante :

  Nom_Matrice = np.array ([[valeur_initiale] * M] * N , Type_élément)

- Les éléments d'une matrice doivent être de **même type**.
- En Python, pour déclarer une matrice de chaînes de caractères, on ajoute "**dtype = object**" à la déclaration ou on supprime les parenthèses après le "**str**".
- En algorithmique, les indices des éléments d'une matrice sont de **type scalaire**.
- Les indices de la 1ère ligne et de la 1ère colonne d'une matrice sont, **par défaut**, égaux à **Zéro**.
- Pour accéder à un élément d'une matrice : **Nom_Matrice [Ligne, Colonne]**

6.3. Les enregistrements

6.3.1. Déclaration en algorithmique

❖ 1ère méthode

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Enregistrement	**Enregistrement** Champ1 : Type1 Champ2, Champ3 : Type2 …… ChampN : TypeM **Fin**

❖ 2ème méthode

Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Nom_Type_Enregistrement = **Enregistrement** Champ1 : Type1 Champ2, Champ3 : Type2 …… ChampN : TypeM **Fin**

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Enregistrement	Nom_Type_Enregistrement

6.3.2. Déclaration en Python

Déclaration en Python	
Nom_Enregistrement = { }	*« signifie Enregistrement vide »*
Nom_Enregistrement = { **Champ1 : Type1,** **Champ2 : Type2,** **ChampN : TypeN** **}**	

6.3.3. Exemple

Déclarer un enregistrement nommé « **Elève** » formé de deux champs : « Nom » et « Age ».

❖ Déclaration en algorithmique (1ère méthode)

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Elève	Enregistrement Nom : Chaîne Age : Entier Fin

❖ Déclaration en algorithmique (2ème méthode)

Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Personne = Enregistrement Nom : Chaine Age : Entier Fin

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Elève	Personne

❖ Déclaration en Python

```
Elève = {
            "Nom " :  str ( ) ,
            "Age " : int ( )
          }
```

N.B. :

- Un champ d'un enregistrement peut être aussi un enregistrement.
- On peut déclarer des tableaux d'enregistrements.
- L'accès à un champ d'un enregistrement se fait de la manière suivante :

Notation en algorithmique	Notation en Python
Nom_Objet . Champ	Nom_Objet ["Champ"]

6.4. Les fichiers

6.4.1. Déclaration en algorithmique (1ère méthode)

	Tableau de Déclaration des Objets (T.D.O)	
	Objet	**Type/Nature**
Fichier texte	Nom_Fichier	Fichier texte
Fichier de données (typé)	Nom_Fichier	Fichier de Type _élément

6.4.2. Déclaration en algorithmique (2ème méthode)

	Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Fichier texte	Nom_Type = Fichier texte
Fichier de données (typé)	Nom_Type = Fichier de Type _élément

	Tableau de Déclaration des Objets (T.D.O)	
	Objet	**Type/Nature**
Fichier texte	Nom_Fichier	Nom_Type
Fichier de données (typé)	Nom_Fichier	Nom_Type

N.B. :

- La position initiale du pointeur dans un fichier texte ou de données est **Zéro**.
- **Type_Element** peut être : Entier, Réel, Caractère, chaîne de caractères, Booléen, Enregistrement.

7. Les structures de contrôle conditionnelles

7.1. La structure conditionnelle simple

Notation en algorithmique	Notation en Python
Si Condition **Alors** Traitement **FinSi**	**if** Condition : Traitement

7.2. La structure conditionnelle complète

Notation en algorithmique	Notation en Python
Si Condition **Alors** Traitement1 **Sinon** Traitement2 **FinSi**	**if** Condition : Traitement1 **else :** Traitement2

7.3. La structure conditionnelle généralisée (Si imbriquée)

Notation en algorithmique	Notation en Python
Si Condition1 **Alors** Traitement1 **Sinon** **Si** Condition2 **Alors** Traitement2 **Sinon** TraitementN **FinSi**	**if** Condition1 : Traitement1 **elif** Condition2 : Traitement2 **else :** TraitementN

7.4. La structure conditionnelle à choix multiples

Notation en algorithmique	Notation en Python (Versions ≤3.9)
Selon Sélecteur Val1 : Traitement1 Val2 , Val3 , Val4 : Traitement2 Val5 .. Val 6 : Traitement3 ..... **Sinon** TraitementN **FinSelon**	Pas de correspondance en Python, toutefois, on pourra utiliser la structure Si généralisée : **if** Sélecteur == Val1 : Traitement1 **elif** Sélecteur **in** { Val2 , Val3 , Val4 } : Traitement2 **elif** Sélecteur **in range** (Val5 , Val6 + 1) : Traitement3 **else :** TraitementN **Remarque** : Les valeurs Val5 et Val6 doivent être des valeurs numériques. Pour les caractères, on doit utiliser le code ASCII.

Notation en algorithmique	Notation en Python (Versions ≥3.10)
Selon Sélecteur Val1 : Traitement1 Val2 , Val3 , Val4 : Traitement2 Val5 .. Val 6 : Traitement3 …… **Sinon** TraitementN **FinSelon**	**match** Sélecteur : **case** Val1 : Traitement1 **case** Val2 | Val3 | Val4 : Traitement2 **case** Sélecteur if Val5<=Sélecteur<=Val6 : Traitement3 **case** _: TraitementN

Exemple

On se propose d'afficher la nature d'un caractère donné (consonne, voyelle, chiffre, symbole). Écrire un algorithme "Nature" correspondant à la résolution de cette situation puis l'implémenter en Python.

```
ALGORITHME Nature
DEBUT
  Ecrire ("Saisir un caractère : ")
  Lire (c)
  c ← Majus (c)
  Selon c
    "A" .. "Z" : Selon c
                   "A", "E", "I", "O", "U", "Y" : Ecrire ("Voyelle")
                   Sinon
                       Ecrire ("Consonne")
                 FinSelon
    "0" .. "9" : Ecrire ("Chiffre")
    Sinon
        Ecrire ("Symbole")
  FinSelon
FIN
```

T.D.O	
Objet	**Type/Nature**
c	Caractère

```
c=input("Saisir un caractère : ")
c=c.upper()
match c :
  case c if "A"<= c <="Z" :
     match c :
        case "O"|"I"|"Y"|"E"|"A"|"U":
              print("Voyelle")
        case _ :
              print("Consonne")
  case c if "0" <= c <= "9":
        print("Chiffre")
  case _ :
        print("Symbole")
```

8. Les structures de contrôle itératives

8.1. La structure de contrôle itérative complète

La boucle Pour … Faire …	
Notation en algorithmique	**Notation en Python**
Pour Compteur de **Début** à **Fin** [Pas= valeur_pas] **Faire** Traitement **FinPour**	**for** Compteur **in range** (Début, Fin+1, Pas) : Traitement
Remarques : • Le nombre de répétitions est connu avant le traitement et il est égal à |**Fin – Début** |+ **1**. • Le **Pas** peut être **Positif** ou **Négatif**. • Par défaut, le **Pas** est égal à **1**. • Il faut éviter de modifier la valeur du compteur de la boucle **Pour… Faire…** au niveau du traitement.	• En python, on accepte seulement la forme décrite précédemment. • **range** (**5**) *le compteur prendra les valeurs suivantes :* ***0 , 1 , 2 , 3 , 4*** • **range** (**2, 5**) *le compteur prendra les valeurs suivantes :* ***2 , 3 , 4*** • **range** (**5, 2, -1**) *le compteur prendra les valeurs suivantes :* ***5 , 4 , 3*** • **range** (**0, 10, 3**) *le compteur prendra les valeurs suivantes :* ***0 , 3 , 6 , 9***

Notation en algorithmique	Notation en Python (Versions ≥3.10)
Selon Sélecteur Val1 : Traitement1 Val2 , Val3 , Val4 : Traitement2 Val5 .. Val 6 : Traitement3 …… **Sinon** TraitementN **FinSelon**	**match** Sélecteur : **case** Val1 : Traitement1 **case** Val2 | Val3 | Val4 : Traitement2 **case** Sélecteur if Val5<=Sélecteur<=Val6 : Traitement3 **case** _: TraitementN

Exemple

On se propose d'afficher la nature d'un caractère donné (consonne, voyelle, chiffre, symbole). Écrire un algorithme "Nature" correspondant à la résolution de cette situation puis l'implémenter en Python.

```
ALGORITHME Nature
DEBUT
  Ecrire ("Saisir un caractère : ")
  Lire (c)
  c ← Majus (c)
  Selon c
    "A" .. "Z" : Selon c
                   "A", "E", "I", "O", "U", "Y" : Ecrire ("Voyelle")
                   Sinon
                       Ecrire ("Consonne")
                 FinSelon
    "0" .. "9" : Ecrire ("Chiffre")
    Sinon
        Ecrire ("Symbole")
  FinSelon
FIN
```

T.D.O	
Objet	**Type/Nature**
c	Caractère

```
c=input("Saisir un caractère : ")
c=c.upper()
match c :
  case c if "A"<= c <="Z" :
     match c :
        case "O"|"I"|"Y"|"E"|"A"|"U":
             print("Voyelle")
        case _ :
             print("Consonne")
  case c if "0" <= c <= "9":
        print("Chiffre")
  case _ :
        print("Symbole")
```

8. Les structures de contrôle itératives

8.1. La structure de contrôle itérative complète

La boucle Pour … Faire …	
Notation en algorithmique	**Notation en Python**
Pour Compteur de **Début** à **Fin** [Pas= valeur_pas] **Faire** Traitement **FinPour**	**for** Compteur **in range** (Début, Fin+1, Pas) : Traitement
Remarques : • Le nombre de répétitions est connu avant le traitement et il est égal à |**Fin – Début** |+ **1**. • Le **Pas** peut être **Positif** ou **Négatif**. • Par défaut, le **Pas** est égal à **1**. • Il faut éviter de modifier la valeur du compteur de la boucle **Pour… Faire…** au niveau du traitement.	• En python, on accepte seulement la forme décrite précédemment. • **range** (**5**) *le compteur prendra les valeurs suivantes :* ***0 , 1 , 2 , 3 , 4*** • **range** (**2, 5**) *le compteur prendra les valeurs suivantes :* ***2 , 3 , 4*** • **range** (**5, 2, -1**) *le compteur prendra les valeurs suivantes :* ***5 , 4 , 3*** • **range** (**0, 10, 3**) *le compteur prendra les valeurs suivantes :* ***0 , 3 , 6 , 9***

8.2. La structure de contrôle itérative à condition d'arrêt (Répéter … Jusqu'à …)

La boucle REPETER … JUSQUA …	
Notation en Algorithmique	**Notation en Python**
Répéter Traitement **Jusqu'à** Condition(s) de sortie	Pas de correspondance. Toutefois, on peut utiliser : valide = **False** **while** valide == **False** : Traitement valide = (Condition(s) de sortie)
Remarque : Le nombre de répétitions n'est pas connu à l'avance et le traitement se fait au moins une fois.	

8.3. La structure de contrôle itérative à condition d'arrêt (Tant que … Faire)

La boucle TANTQUE … FAIRE …	
Notation en Algorithmique	**Notation en Python**
Tantque Condition **Faire** Traitements **FinTantque**	**while** Condition : Traitements
Remarque : Le nombre de répétitions n'est pas connu à l'avance et le traitement peut ne pas se faire.	

N. B. :

- En Python, il est conseillé d'éviter l'utilisation de l'instruction « **break** », « **continue** » et « **pass** » dans les structures conditionnelles et les structures itératives.
- En algorithmique, la structure itérative « **Répéter… Jusqu'à…** » doit être enseignée bien qu'elle n'a pas de correspondance en Python.

9. Les modules

9.1. Les fonctions

9.1.1. La définition d'une fonction

Notation algorithmique

```
FONCTION Nom_fonction ( Pf1 : Type1 , … , Pfn : Typen ) : Type_Résultat
DEBUT
        Instruction1
        InstructionN
        Retourner  Résultat
FIN
```

Notation en Python

```
def  Nom_fonction ( Pf1 , … , Pfn ) :
     Instruction1
     InstructionN
     return  Résultat
```

9.1.2. L'appel d'une fonction

Notation algorithmique

Objet ← ***Nom_fonction*** (Pe_1 , … , Pe_n)

Notation en Python

Objet = ***Nom_fonction*** (Pe_1 , … , Pe_n)

N. B. :

- La fonction retourne un **seul résultat** (Entier, Réel, Booléen, Caractère ou Chaîne de caractères).
- Les paramètres effectifs (Pe_1 à Pe_n) et les paramètres formels (Pf_1 à Pf_n) doivent s'accorder de point de vue **ordre**, **nombre** et **type**.
- L'appel d'une fonction est une **expression**.

9.2. Les procédures

9.2.1. La définition d'une procédure

Notation algorithmique

PROCEDURE ***Nom_procédure*** (**Pf_1 : $Type_1$** , **Pf_2 : $Type_2$** , … , **Pf_n : $Type_n$**)
DEBUT
    Instruction1
    Instruction2
    InstructionN
FIN

Notation en Python

def ***Nom_procédure*** (Pf_1 , … , Pf_n) :
    Instruction1
    Instruction2
    InstructionN

9.2.2. L'appel d'une procédure

Notation algorithmique

Nom_procédure (Pe_1 , … , Pe_n)

Notation en Python

Nom_procédure (Pe_1 , … , Pe_n)

N. B. :

- Les paramètres effectifs (Pe_1 à Pe_n) et les paramètres formels (Pf_1 à Pf_n) doivent s'accorder de point de vue **ordre**, **nombre** et **type**.
- L'appel d'une procédure est une **instruction**.
- Le passage de paramètre par **adresse (par référence)** permet au programme appelant (**PP**) de transmettre une valeur à la procédure appelée (**SP**) et **vice versa**. Le changement du paramètre formel permet aussi le changement du paramètre effectif. On ajoutera le symbole « @ » avant le paramètre formel passé par adresse.

- Si nous avons plusieurs paramètres de même type et qui ont un passage par adresse, ils doivent être précédés par « @ ». Par exemple : **PROCEDURE** Traitement (@ A, B : Entier, X, Y : Réel)
- En Python, pour résoudre le problème de passage par adresse, on peut suivre l'une des deux démarches suivantes :
  - **La 1ère démarche :**
    1°) Ne pas mettre les paramètres formels passés par adresse dans l'entête de la procédure.
    2°) Mettre les paramètres formels passés par adresse dans le corps de la procédure précédés du mot « **global** ».
  - **La 2ème démarche :**
    1°) Ne pas mettre les paramètres formels passés par adresse dans l'entête de la procédure.
    2°) Utiliser le mot « **return** » pour retourner les valeurs des paramètres formels passés par adresse (au niveau algorithme).

L'exemple ci-après illustre le passage par adresse en algorithmique et en Python.

Notation en algorithmique

Déclaration de la procédure "Saisir"	**L'appel de la procédure "Saisir"**

Procédure Saisir (@ n : entier)
Début
  Répéter
  Écrire ("Saisir un entier entre 5 et 20 : ")
  Lire (n)
  Jusqu'à (5 ≤ n ≤ 20)
Fin
	Saisir (n)

Notation en Python de la 1ère démarche

Déclaration de la procédure "Saisir"	**L'appel de la procédure "Saisir"**

def Saisir ( ) :
  global n
  valid = False
  while valid == False :
  n = int (input("Saisir un entier entre 5 et 20 : "))
  valid = (5 ≤ n ≤ 20)
	Saisir ( )

Notation en Python de la 2ème démarche

Déclaration de la procédure "Saisir"	**L'appel de la procédure "Saisir"**

def Saisir ( ) :
  valid = False
  while valid == False :
  n = int (input("Saisir un entier entre 5 et 20 : "))
  valid = (5 ≤ n ≤ 20)
  return n
	n = Saisir ( )

Exemple

Objectif : Remplir puis afficher un tableau t par n entiers donnés avec ($5 \leq n \leq 10$).

En algorithmique

Procédure Saisir (@ m : entier)
Début
    Répéter
        Ecrire ("Taille du tableau : ")
        Lire (m)
    Jusqu'à m ∈ [5..10]
Fin

Procédure Remplir (@v : tab, m : entier)
Début
    Pour i de 0 à m-1 faire
        Ecrire ("T[",i,"] = ")
        Lire(v[i])
    FinPour
Fin

T.D.O.L	
Objet	**Type/Nature**
i	Entier

Procédure Afficher (v : tab, m : entier)
Début
    Pour i de 0 à m-1 faire
        Écrire(v[i])
    FinPour
Fin

T.D.O.L	
Objet	**Type/Nature**
i	Entier

#Algorithme du programme principal
ALGORITHME Exemple_Tableau
DÉBUT
    Saisir (n)
    Remplir (t, n)
    Afficher (t, n)
FIN

T.D.N.T
Type
tab = tableau de 10 entiers

T.D.O	
Objet	**Type/Nature**
n	entier
t	tab
Saisir	Procédure
Remplir	Procédure
Afficher	Procédure

En Python

```
from numpy import array

def Saisir() :
        valide= False
        while valide == False :
            m= int(input("Taille du tableau : "))
            valide = m in range(5,11)
        return m

def Remplir(m) :
        v = array([int()] * m)
        for i in range(m):
            v[i]=int(input("T["+str(i)+"] = "))
        return v

def Afficher(v,m) :
        for i in range(m):
            print(v[i],end="  ")

# Programme principal
n=Saisir()
t=Remplir(n)
Afficher(t,n)
```

Remarque : On peut développer un seul module pour saisir la taille du tableau et pour remplir le tableau par N entiers. La solution sera :

En algorithmique

Procédure Remplir (@v : tab, @ m : entier)
Début
    Répéter
        Ecrire ("Taille du tableau : ")
        Lire(m)
    Jusqu'à (m ≥5) et (m ≤ 10)
    Pour i de 0 à m-1 faire
        Ecrire ("T[",i,"] = ")
        Lire(v[i])
    Fin
Fin

T.D.O.L	
Objet	**Type/Nature**
i	Entier

#Algorithme du programme principal

ALGORITHME Exemple_Tableau
DÉBUT
    Remplir (t, n)
    Pour i de 0 à n-1 faire
        Écrire(t[i])
    FinPour
FIN

T.D.N.T
Type
Tab = tableau de 10 entiers

T.D.O	
Objet	**Type/Nature**
n	Entier
t	Tab
i	Entier
Remplir	Procédure

En Python

```
from numpy import array

def Remplir() :
        valide=False
        while valide == False :
            m= int(input("Taille du tableau : "))
            valide = m in range(5,11)
        v = array([int()] * m)
        for i in range(m):
            v[i]=int(input("T["+str(i)+"] = "))
        return v,m

# Programme principal
t,n=Remplir()
for i in range(n):
            print(t[i],end="  ")
```

10. Les fonctions et les procédures sur les fichiers

10.1. Les fonctions et les procédures sur les fichiers de données (typés)

<table>
<tr><th colspan="2">Notation algorithmique</th><th>Rôle</th></tr>
<tr><td colspan="2">Ouvrir ("Chemin\Nom_Fhysique" , Nom_Logique , "Mode")</td><td rowspan="2">Ouverture d'un fichier.

Mode d'ouverture :
• "rb" : Lecture
• "wb" : Écriture (Création)
• "ab" : Écriture à la fin du fichier</td></tr>
<tr><td colspan="2">Notation en Python
Nom_Logique = open ("Chemin\Nom_Physique" , "Mode")
ou bien
Nom_Logique = open ("Chemin/Nom_Physique" , "Mode")</td></tr>
<tr><td>Notation algorithmique</td><td>Notation en Python</td><td>Rôle</td></tr>
<tr><td>Lire (Nom_Logique , Objet)</td><td>import pickle as pic
Objet = pic.load (Nom_Logique)</td><td>Lecture d'un enregistrement d'un fichier.</td></tr>
<tr><td>Ecrire (Nom_Logique , Objet)</td><td>import pickle as pic
pic.dump (Objet , Nom_Logique)</td><td>Écriture dans un fichier.</td></tr>
<tr><td>Fin_Fichier (Nom_Logique)</td><td>Pas de correspondance</td><td>Retourner VRAI si le pointeur est à la fin du fichier sinon elle retourne FAUX.</td></tr>
<tr><td>Fermer (Nom_Logique)</td><td>Nom_Logique.close ( )</td><td>Fermeture du fichier.</td></tr>
</table>

N.B. :

- La position initiale du pointeur dans un fichier de données est **Zéro**.
- Les traitements sont réalisés dans la mémoire centrale.

Exemple

Objectif : Remplir un fichier typé par les noms et les moyennes de N élèves avec (2≤N≤20) puis afficher la moyenne de la classe et la liste des noms des élèves admis (Moyenne >= 10).

En algorithmique

Procédure Saisir (@ n : entier)
Début
  Répéter
    Ecrire ("Saisir le nombre d'élèves : ")
    Lire (n)
  Jusqu'à n ∈ [2..20]
Fin

Procédure Remplissage_Fichier (nomphy : chaîne, n : entier)
Début
  Ouvrir (nomphy, f , "wb")
  Pour i de 0 à n-1 Faire
    Ecrire ("Saisir le nom de l'élève N° ", i), Lire (e.nom)
    Ecrire ("Saisir la moyenne de l'élève N° ', i), Lire (e.moy)
    Ecrire (f, e)
  FinPour
  Fermer (f)
Fin

T.D.N.T	
Type	
Eleve = Enregistrement	
	nom : chaine
	moy : Réel
Fin	

T.D.O	
Objet	**Type/Nature**
i	Entier
e	Eleve
f	Fichier d'Eleve

Fonction Moyenne (nomphy : chaîne, n : entier) : réel
Début
  Ouvrir (nomphy, f , "rb")
  s ← 0
  Pour i de 0 à n-1 Faire
    Lire (f, e)
    s ← s + e.moy
  FinPour
  Fermer (f)
  Retourner (s/n)
Fin

T.D.N.T	
Type	
Eleve=Enregistrement	
	nom : chaine
	moy : Réel
Fin	

T.D.O	
Objet	**Type/Nature**
i	Entier
e	Eleve
s	Réel
f	Fichier d'Eleve

En Python

```
# Importation de la bibliothèque pickle
import pickle as pic

# Procédure Saisir
def Saisir ( ) :
    global n
    valide = False
    while (valide == False) :
        n = int (input("Saisir le nombre d'élèves : "))
        valide = (2<=n<=20)

# Procédure Remplissage_Fichier
def Remplissage_Fichier (nomphy, n) :
    f = open (nomphy, "wb")
    for i in range (0, n) :
        e = { "nom":"", "moy":0 }
        e["nom"]= input ("Saisir le nom de l'élève N°"+ str(i+1) +":")
        e["moy"]= float (input( "Saisir la moyenne de l'élève N°"+ str(i+1)+" "))
        pic.dump (e , f)
    f.close()

# Fonction Moyenne
def Moyenne (nomphy, n) :
    f = open (nomphy, "rb")
    s = 0
    for i in range (0 , n) :
            e= pic.load ( f )
            s= s + e ["moy"]
    f.close ()
    return s/n
```

Procédure Affichage (nomphy : chaîne)
Début
    Ouvrir (nomphy, f , "rb")
    Ecrire_nl ("La liste des admis : ")
    Tantque Non Fin_Fichier (f) Faire
        Lire (f, e)
        Si e.moy ≥ 10 Alors
            Ecrire_nl (e.nom)
        FinSi
    FinTantque
    Fermer (f)
Fin

T.D.N.T
Type
Eleve = Enregistrement nom : chaîne moy : Réel Fin

T.D.O	
Objet	Type/Nature
e	Eleve
f	Fichier de Eleve

#Algorithme du programme principal
ALGORITHME Admis
DÉBUT
    nomphy ← "c:\bac2022\eleves.dat"
    Saisir (n)
    Remplissage_Fichier (nomphysique, n)
    Ecrire_nl ("La moyenne de la classe est : ", Moyenne (nomphy , n))
    Affichage (nomphy)
FIN

T.D.O	
Objet	Type/Nature
nomphy	Chaîne
n	Entier
Saisir	Procédure
Remplissage_Fichier	Procédure
Moyenne	Fonction
Affichage	Procédure

```
# Procédure Affichage
def Affichage (nomphy) :
    f = open (nomphy, "rb")
    print ("La liste des admis : ")
    fin_fichier = False
    while fin_fichier  == False :
        try :
            e= pic load (f)
            if (e["moy"] >= 10) :
                print (e ["nom"])
        except :
            fin_fichier = True
    f.close ( )


# Le programme principal
nomphy = "c:\bac2022\eleves.dat"
Saisir ( )
Remplissage_Fichier (nomphy, n)
print ("La moyenne de la classe est : ", Moyenne (nomphy , n))
Affichage (nomphy)
```

10.1. Les fonctions et les procédures sur les fichiers Texte

Notation algorithmique		Rôle
Ouvrir ("Chemin\Nom_Physique" , *Nom_Logique* , "Mode")		Ouverture d'un fichier. **Mode d'ouverture** : • "**r**" : Lecture • "**w**" : Ecriture (Création) • "**a**" : Ajout à la fin du fichier
Notation en Python		
Nom_Logique = **open** ("Chemin\Nom_Physique", "Mode") *ou bien* *Nom_Logique* = **open** ("Chemin/Nom_Physique", "Mode")		
Notation algorithmique	**Notation en Python**	**Rôle**
Lire (*Nom_Logique* , Ch)	Nom_Logique = **open** ("chemin\Nom_Physique" , "r") Ch = *Nom_Logique*.**read** ( ) *ou bien* T = *Nom_Logique*.**readlines** ( )	Lecture de la totalité d'un fichier.
Lire_Ligne (*Nom_Logique* , Ch)	*Nom_Logique* = **open** ("chemin\Nom_fichier.extension" , "r") Ch = *Nom_Logique*.**readline** ( )	Lecture d'une ligne depuis un fichier texte.
Ecrire (*Nom_Logique* , Ch)	*Nom_Logique* = **open** ("chemin\Nom_fichier.txt" , "w ou a") *Nom_Logique*.**write** (Ch)	Écriture de la chaîne Ch dans un fichier texte.
Ecrire_nl (*Nom_Logique* , Ch)	*Nom_Logique* = **open** ("chemin\Nom_fichier.txt" , "w ou a") *Nom_Logique*.**write** (Ch + "\ n")	Écriture de la chaîne Ch dans un fichier texte et retour à une nouvelle ligne
Fin_Fichier (*Nom_Logique*)	Pas de correspondance	Retourner **VRAI** si le pointeur est à la fin du fichier Sinon elle retourne **FAUX**
Fermer (*Nom_Logique*)	*Nom_Logique*.**close** ( )	Fermeture du fichier

N.B. :

- La position initiale du pointeur dans un fichier texte est **Zéro**.
- En python, lors de la lecture d'une chaîne à partir d'un fichier, on utilise la methode « **rstrip** ( ) » pour supprimer le retour à la ligne existant par défaut à la fin de cette chaîne.

Exemple

Objectif : Remplir un fichier texte par des chaines de caractères non vides puis calculer et afficher le nombre de mots dans ce fichier. Le remplissage du fichier s'arrête lorsque l'utilisateur répond par « **N** » à la question « **Voulez-vous continuer la saisie O / N ?** »

En algorithmique

```
Procédure Remplissage_Fichier (nomphy : chaîne)
Début
    Ouvrir (nomphy, f , "w")
    Répéter
        Répéter
            Ecrire ("Saisir votre chaîne : "), Line (ch)
        Jusqu'à ch ≠ ""
        Ecrire_al (f, ch)
        Ecrire ("Voulez-vous continuer la saisie O / N ? ")
        Lire (rep)
    Jusqu'à (majus (rep)= "N")
    Fermer (f)
Fin
```

T.D.O	
Objet	Type/Nature
ch	chaîne
rep	chaîne
f	Fichier Texte

```
Fonction Nettoyage (ch : chaîne) : chaîne
Début
    Tantque (pos("␣␣", ch) ≠ -1 ) Faire          # Le symbole ␣ signifie espace
        p ← pos("␣␣", ch)
        ch ← effacer (ch, p, p+1)
    FinTantque
    Si ch[0] = "␣" Alors
        ch ← effacer (ch, 0, 1)
    FinSi
    Si ch [long(ch) -1 ] = "␣" Alors
        ch ← effacer (ch, long( ch )-1 , long( ch ))
    FinSi
    Retourner ch
Fin
```

T.D.O	
Objet	Type/Nature
ch	chaîne
rep	chaîne
f	Fichier Texte

En Python

```
# Procédure Remplissage_Fichier pour remplir un fichier par des chaînes non vides
def Remplissage_Fichier (nomphy) :
    f = open (nomphy, "w")
    stop=False
    while (stop == False) :
        valid=False
        while valid == False :
            ch = input("Saisir une phrase : ")
            valid= (ch != "")
        f.write (ch+"\n")
        rep = input("Voulez-vous continuer la saisie O/N ? ")
        stop = rep.upper( ) == "N"
    f.close( )

# Fonction Nettoyage pour supprimer les espaces supplémentaires et les espaces au début et à la fin d'une chaîne
def Nettoyage (ch) :
    while (ch.find("  ") != -1) :
        p = ch.find ("  ")
        ch = ch[ :p ] + ch[ p+1 : ]
    if (ch[len(ch)-1] == " ") :
        ch = ch[ :len(ch)-1 ]
    if (ch[0] == " ") :
        ch=ch [ 1: ]
    return ch
```

Fonction Nombre_mot (ch : chaine) : entier

Début *# Le symbole ⊔ signifie espace*

nbm ← 1

Pour i de 0 à long(ch) -1 Faire

Si ch [i] = "⊔" Alors

nbm ← nbm +1

FinSi

FinPour

Retourner nbm

Fin

T.D.O	
Objet	**Type/Nature**
i	Entier
nbm	Entier

Fonction Comptage (nomphy : chaine) : entier

Début

Ouvrir (nomphy, f , "r")

nbmot ← 0

Tantque Non Fin_Fichier (f) Faire

Lire_nl (f, ch)

ch1 ← Nettoyage (ch)

nbmot ← nbmot + Nombre_mot (ch1)

FinTantque

Fermer (f)

Retourner Nbmot

Fin

T.D.O	
Objet	**Type/Nature**
f	Fichier Texte
ch	Chaine
ch1	Chaine
nbmot	Entier
Nettoyage	Fonction
Nombre_Mot	Fonction

 # Fonction Nombre_mot pour calculer le nombre de mots dans une chaine. Sachant que nombre de mots = nombre d'espaces + 1

```
def Nombre_mot (ch) :
    nbm = 1
    for i in range (0, len (ch)-1) :
        if ch[i] ==" " :
            nbm = nbm + 1
    return nbm
```

 # Fonction Comptage pour calculer le nombre de mots dans le fichier

```
def Comptage (Nomphy) :
    f = open(nomphy , "r")
    nbmot = 0
    ch = f.readline()
    while ch != "" :
        ch1 = Nettoyage(ch.rstrip( ))
        nbmot= nbmot + Nombre_mot (ch1)
        ch=f.readline ( )
    f.close ( )
    return nbmot
```


#Algorithme du programme principal

ALGORITHME Mot

DÉBUT

nomphy ← "C:\Bac2022\phrases.txt"

Remplissage_Fichier (nomphy)

Ecrire ("Le nombre de mots dans le fichier = ", comptage (nomphy))

FIN

T.D.O	
Objet	**Type/Nature**
nomphy	chaine
Remplissage_Fichier	Procédure
Comptage	Fonction
Affichage	Procédure

 # Le programme principal

```
nomphy="C:\Bac2022\phrases.txt"
Remplissage_Fichier (nomphy)
print ("Le nombre de mots dans le fichier = ", Comptage (nomphy))
```