I-Opérateurs & Fonctions Standard

Operateurs arithmétiques		
Opération	Opérateur en algorithme	Opérateur en python
Somme	+	+
Soustraction	-	-
Multiplication	*	*
Division	/	/
Division entière	Div	//
Reste de la division entière	Mod	%

Opérateurs de comparaison		
Opération	Opérateur en algorithme	Opérateur en python
Egal	=	==
Différent	≠	!=
Strictement supérieur	>	>
Supérieur ou égal	≥	>=
Strictement inférieur	<	<
Inférieur ou égal	≤	<=
Appartient(entier ou caractére)	ϵ	In

Opérateurs logiques		
Opération	Opérateur en algorithme	Opérateur en python
Négation	Non	Not
Conjonction	Et	And
Disjonction	Ou	Or

II-Les fonctions prédéfinies :

Fonctions arithmétiques standards		
En algorithme	En python	Rôle
Arrondi(d)	round(d)	Retourne un nombre qui est la valeur de d arrondie à la plus proche valeur
Racine carré (d)	math.sqrt(d)	Si d est positif elle retourne sa racine carré sinon elle provoque une erreur
Aléa(vi,vf)	randint(vi,vf)	Retourne un entier aléatoire dans l'intervalle [vi,vf]
Ent (x)	int(x)	Retourne la partie entière de x.
Abs(x)	ABS(x)	Retourne la valeur absolue de x

Fonctions standards sur les caractères		
En algorithme	En python	Rôle
Ord(c)	Ord (c)	Retourne le code ASCII du caractère c
Chr(n)	Chr(n)	Retourne le caractère dont le code ASCII est c

Fonctions standards sur les chaines		
En algorithme	En python	rôle
+	+	Permet la concaténation d'un ensemble des chaines de caractères
Long(ch)	len(d)	Retourne le nombre de caractères de la chaine ch
Pos(ch1,ch2)	ch2.find(ch1)	Retourne la première position de la chaine ch1 dans ch2
Convch(d)	str(d)	Retourne la conversion de d en chaines de caractères
Valeur(ch)	float (ch) int (ch)	Retourne la conversion de la chaine ch en une valeur numérique sinon elle provoque une erreur
Sous_chaîne(ch,d,f)	ch[d :f]	Retourne une partie de la chaîne ch à partir de la position **d** jusqu'à la position **f** (f exclue).
Effacer (ch, d, f)	ch[:d]+ch[f+1:]	Efface des caractères de la chaîne ch à partir de la position **d** jusqu'à la position **f** (f exclue).
Majus (ch)	ch.upper()	Convertit la chaîne ch en majuscules.
Estnum (ch)	ch.isdigit()	Retourne **Vrai** si la chaîne ch est convertible en une valeur numérique, elle retourne Faux sinon.

III. Les Types de données :

En algorithmique	En algorithmique
Entier	Int
Réel	Float
Booléen	Bool
Chaine de caractère/caractère	Str
Tableau	Array

Exemples de conversions entre les types simples en python

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

IV. Les instructions simples

1. Les structures Simples :

a. L'opération d'entrée

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

b. L'opération de sortie

En algorithmique	En python
Écrire ("Message", Objet, Expression)	**print** ("Message", Objet, Expression)
Écrire_nl ("Message", Objet, Expression)	**print** ("Message", Objet, Expression, "\n")

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).

V. Les structures de contrôle :

1-Les structures de contrôles conditionnelles :

	En algorithmique	En python
La forme simple	**SI** condition **ALORS** **Traitement1** **SINON** **Traitement2** **FIN SI**	**If** condition **:** **Traitement1** **else :** **Traitement2**
La forme généralisée	**SI Condition1ALORS Traitement 1** **SINON SI Condition 2ALORS Traitement 2** **SINON SI Condition 3ALORS Traitement 3** **SINON SI Condition 4 ALORS Traitement 4** **……..…** **SINON SI condition n-1 ALORS Traitement n-1** **SINON Traitement n** **FIN SI**	**If Condition 1 : Traitement1** **elif Condition 2 : Traitement 2** **elifCondition 3 : Traitement3** **elifCondition 4 : Traitement4** **……..…** **elifCondition n-1 :Traitement n-1** **else : Traitement n**
La forme à choix	**Selon** <Sélecteur> Valeur1_1[, Valeur1_2, …] : Traitement1 Valeur2_1 . . Valeur2_2 : Traitement2 …… [**Sinon** TraitementN] **Fin Selon**	**match** Sélecteur : **case** Valeur1 **:** Traitement1 **case** Valeur2_1 | Valeur2_2 **:** Traitement2 **case** Sélecteur if V3_1 <=Sélecteur<= V3_2 **:** Traitement3 **case** _ **:** TraitementN

2. Les structures de contrôles itératives :

	En algorithmique	En python
La boucle Pour	**Pour compteur de Vi à Vf (pas) Faire** Traitement **Fin Pour**	**for compteur in range (Vi, Vf, pas) :** Traitement
La boucle répéter	**Répéter** Traitement **Jusqu'à condition d'arrêt**	**While** *condition* **:** Traitement
La boucle Tant que	**Tant que** *condition de continuité* **faire** Traitement **Fin tant que**	**While** *condition de continuité* **:** Traitement

VI. Les types avancés :

	Nouveau type
Tableau à une dimension	**Nom_type** = Tableau de N Type _élément
Tableau à deux dimensions	**Nom_type** = Tableau de N lignes * M colonnes Type _élément
Enregistrement	**Nom_type** = Enregistrement Nom_champ1 : Type_champ1 Nom_champ2 : Type_champ2 ... Fin

La déclaration d'un tableau se fait en deux étapes :

- **Importation** des modules nécessaires de la bibliothèque **numpy**

Importation
from numpy import array ou **from numpy import *** ou **import numpy as alias**

- **Déclaration** du tableau

	Déclaration
Tableau à une dimension	T = **array ([**Type_élément**]** * N**)** **ou bien** T = **array ([**valeur_initiale**]** * N**)**
Tableau à deux dimensions	T = **array ([**Type_élément**]***Colonnes**]*** Lignes**)** **ou bien** T = **array ([**valeur_initiale**]***Colonnes**]*** Lignes**)**

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

Nom_tableau **= *array ([****Valeur_initiale****]*** * *N****, dtype****=****Type_élément****)***

En Python
Nom_enregistrement = dict (**Nom_champ1 = Type_champ1,** **Nom_champ2 = Type_champ2,** **...** **)**

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

Exemples de déclarations de tableaux en Python

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

VIII. Les modules :

a. La déclaration

En algorithmique	En Python
Fonction Nom_fonction (pf_1 : $type_1$, pf_2 : $type_2$, … , pf_n : $type_n$) **: Type_résultat** **DEBUT** Traitement **Retourner** résultat **FIN**	Un module (fonction ou procédure) se définit en utilisant le mot clé **def** selon la syntaxe suivante : **def Nom_module (pf_1 , pf_2 , … , pf_n) :** **Traitement** **[return résultat]**
Procédure Nom_procédure (pf_1 : $type_1$, pf_2 : $type_2$, … , pf_n : $type_n$) **DEBUT** Traitement **FIN**	***N.B.*** **:** Dans un module, l'instruction "**return**" peut être utilisée, et ce, pour retourner **un seul résultat de type simple**.

b. L'appel

Module	En algorithmique	En Python
Fonction	Objet ←Nom_fonction (pe_1 , …, pe_n)	**Objet = Nom_module (pe_1 , …, pe_n)**
Procédure	Nom_procédure (pe_1 , … , pe_n)	**Nom_module (pe_1 , … , pe_n)**

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. **Procédure** Nom_procédure (@pf_1 : $type_1$, @pf_2 : $type_2$, … , pf_n : $type_n$) **DEBUT** Traitement **FIN**	Nom_module (pf_1 , pf_2 , … , pf_n) : Traitement ***N.B.*** **:** En python, les paramètres de type **dictionnaire**, **tableau** et **fichier** sont, par défaut passés par **référence**.