Ce document est un aide-mémoire sur la façon d’écrire les algorithmes ainsi que leurs équivalents en langage Python.

|  |
| --- |
| Variables et données : |

## Déclaration de variable :

Les variables sont déclarées en en-tête dans une partie dédiée. Le Python n’étant pas un langage fortement typé, il ne faut pas déclarer de variables en Python.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Booléen | b : booléens |  |
| Entier | e : entier |  |
| Réel | R : réel |  |
| Chaîne de caractères | C : chaîne de caractères |  |

## Les tableaux

Les tableaux peuvent être déclarés en spécifiant leur type et leur taille ou en donnant la liste de leurs éléments en plus du type.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Tableau de 30 éléments | T : entier[30] | T = [0] \* 30 |
| Tableau contenant les élément 15, 5 et 17 | t : entier[] = [15, 5, 17] | T = [15, 5, 17] |

## Affecter des valeurs

Pour pouvoir affecter une valeur à une variable, il faut que la valeur soit du même type que la variable.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Booléen | b 🡨 VRAI b 🡨 FAUX | b = Trueb = False |
| Entier | e 🡨 3 | e = 3 |
| Réel | r 🡨 2.0 | R = 2.0 |
| Chaîne de caractères | C 🡨 ‘Bonjour’ | C = ‘Bonjour’ |
| Tableau/Liste | T 🡨 [15, 5, 17] | T = [15, 5, 17] |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opérations sur les nombres Il est possible de faire des opérations sur les nombres si les variables sont bien du type entier ou réel.   |  |  |  | | --- | --- | --- | |  | Algorithmie | Python | | Addition | C 🡨 a + b | C = a + b | | Soustraction | C 🡨 a – b | C = a – b | | Multiplication | C 🡨 a \* b | C = a \* b | | Puissance | C 🡨 a^b | C = a\*\*b | | Division décimale | C 🡨 a/b | C = a/b | | Quotient de la division | C 🡨 a //b | C= a//b | | Reste de la division | C 🡨 a % b | C = a % b | |
| Comparer Il est possible de comparer deux variables ou une variable et une constante si les deux parties du test sont bien du même type.   |  |  |  | | --- | --- | --- | |  | Algorithmie | Python | | Egalité | A == b | A == b | | Différence | A b | A != b | | Inférieur | A < b | A < b | | Inférieur ou égal | A b | A <= b | | Supérieur | A > b | A > b | | Supérieur ou égal | A | A >= b | |
| Opération sur les booléens Les valeurs booléennes peuvent utiliser les opérateurs logiques.   |  |  |  | | --- | --- | --- | |  | Algorithmie | Python | | Négation | Non A | Not(A) | | Et | a et b | a and b | | Ou | a ou b | a or b | |
| Opération sur les chaînes de caractères Il est possible de manipuler les chaînes de caractères. Les chaînes de caractères peuvent être vues comme des tableaux de lettres et partagent donc des similitudes avec les tableaux.   |  |  |  | | --- | --- | --- | |  | Algorithmie | Python | | Concaténation | Ch1+ch2 | Ch1+ch2 | | Taille | Longueur(ch1) | Len(ch1) | | Premier caractère | Ch1[0] | Ch1[0] | | Caractère d’indice i | Ch1[i] | Ch1[i] | | Sous-chaîne | Ch1[i :j] | Ch1[i :j] | |
| Opération sur les tableaux/listes Le tableau ou la liste en Python est un type qui regroupe plusieurs valeurs du même type.   |  |  |  | | --- | --- | --- | |  | Algorithmie | Python | | Taille | Longueur(l1) | Len(l1) | | Premier élément | l1[0] | l1[0] | | Elément d’indice i | l1[i] | l1[i] | | Sous-liste | l1[i :j] | l1[i :j] | | Ajouter un élément en fin de liste | Ajouter e à l1 | L1.append(e) | |
|  |

|  |
| --- |
| Structure d’un programme : |

## Déclaration de variable :

Un programme contient en-tête la déclaration des variables (nom et type). Puis il donne toutes les instructions qui le compose.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Forme Générique | **Variables :**  var1 : type  var2 : type **Début**  <instructions> **Fin** | En python il n’y a pas de déclaration explicite des variables. |
| Programme d’addition | **Variables :**  a : entier  b : entierC : entier **Début**  a 🡨 3  b 🡨 5  c 🡨 a + b **Fin** | a = 3 b = 5 c = a + b |

## Lecture/écriture des données :

Pour que l’utilisateur puisse interagir avec le programme il faut lui laisser l’occasion de saisir ses données et d’être informé des résultats.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Afficher | **Afficher** ‘Message’ | Print(‘message’) |
| Lire une chaîne de caractères | **Saisir** c | C = input() |
| Lire un entier | **Saisir** e | E = int(input()) |
| Lire un réel | **Saisir** r | R = float(input()) |
| Lire et afficher | **Variables :**  age : entier  nom : chaîne de caractères **Début**  saisir ageSaisir nomAfficher ‘vous vous appelez ‘ + nomAfficher ‘Vous avez ‘ + age + ‘ ans’ **Fin** | Age = int(input())Nom = input()Print(‘Vous vous appelez ‘, nom)Print(‘Vous avez ’, age, ‘ ans’) |

## Instructions conditionnelles :

Les instructions conditionnelles permettent de faire des choix dans nos programmes. En python, c’est l’indentation qui va faire que l’on se retrouve dans un bloc ou non.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Si classique | **Si** < condition> **alors** <instruction> **FinSi** | If <condition> :  <instruction> |
| Si avec clause sinon | **Si** < condition> **alors** <instruction>**Sinon**<instruction> **FinSi** | If <condition> :  <instruction> else :  <instruction> |
| Si avec sinon si | **Si** < condition> **alors** <instruction>**Sinon si** <condition> **alors**<instruction> **FinSi** | If <condition> :  <instruction> elif <condition> :  <instruction> |
| Si avec sinon si et sinon | **Si** < condition> **alors** <instruction>**Sinon si** <condition> **alors**<instruction>**Sinon**<instruction> **FinSi** | If <condition> :  <instruction> elif <condition> :  <instruction> Else :  <instruction> |
| Si imbriqué | **Si** < condition> **alors  Si** < condition> **alors** <instruction> **FinSi FinSi** | If <condition> :  if <condition> :  <instruction> |

Voyons un petit exemple pour illustrer ce contenu :

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Parité | **Variables :**  n : entier **Début**  saisir n**Si** n%2 == 0 **alors**  Afficher ‘Nb pair’  **sinon**  Afficher ‘Nb impair’  **finSi** **Fin** | N = int(input())If n %2 == 0 :Print(‘Nb pair’)Else :Print(‘Nb impair’) |

## Instructions itératives :

Les boucles peuvent prendre plusieurs aspects tant en algorithmie qu’en Python. Il Faut privilégier les boucles pour quand vous savez à l’avance combien d’itération vous allez faire et rester sur la boucle tant que quand la seule chose que vous connaissez est la condition d’arrêt.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Boucle pour avec indice | **Pour** i de 0 à n **faire** <instruction> **FinPour** | For i in range(0, n+1) :  <instruction> |
| Boucle pour sur un tableau | **Pour** e dans l **Faire** <instruction> **FinPour** | For e in l : <instruction> |
| Boucle tant que | **Tantque** <condition> **faire** <instruction> **FinTantque** | While <condition> :  <instruction> |
| Boucle Faire… tant que | **Faire**<instruction>**Tantque** <condition> | N’existe pas en Python |

Voyons un petit exemple pour illustrer ce contenu :

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Afficher les nombres pairs entre 0 et 1000 avec une boucle pour | **Variables :**  n : entier **Début**  n 🡨 1000 **Pour** i de 0 à n **Faire** **Si** i%2 == 0 **alors**  Afficher n  **finSi** **FinPour** **Fin** | n = 1000For i in range(0, n+1) :If i %2 == 0 :Print(n) |
| Afficher les nombres pairs entre 0 et 1000 avec une boucle tantque | **Variables** :  n : entierI : entier **Début**  n 🡨 1000I 🡨 0 **Tantque** i <= n **Faire****Si** i%2 == 0 **alors**  Afficher n  **finSi**I 🡨 i + 1**FinTanque Fin** | n = 1000i = 0while i <= nIf i %2 == 0 :Print(n)I = i + 1 |

# Fonctions :

Les fonctions vont nous permettre de factoriser le code de notre programme et d’appeler un même code plusieurs fois.

## Signatures et prototypes :

La signature ou prototype de la fonction, nous permet de connaitre :

* Le nom de la fonction
* Les paramètres de la fonction
* La valeur de retour de la fonction s’il y en a une

En Python la valeur de retour n’est pas présentée dans l’en-tête.

|  |  |  |
| --- | --- | --- |
|  | Algorithmie | Python |
| Fonction sans paramètre et sans valeur de retour. | **Fonction** fct() : rien**Variables locales :**<déclaration>**Début**<instruction>**Fin** | Def fct() :<instruction> |
| Fonction sans paramètre avec valeur de retour | **Fonction** fct() : entier**Variables locales :**E : entier**Début**<instruction> **Retourner(**E**)****Fin** | Def fct() :<instruction>Return(E) |
| Fonction avec paramètres avec valeur de retour | **Fonction** fct(n1 : entier, n2 : entier) : entier**Variables locales :**E : entier**Début**<instruction> **Retourner(**E + n1 + n2**)****Fin** | Def fct(n1, n2) :<instruction>Return(E+n1+n2) |