Algorithmie

Algorithmie : méthode résolution de problèmes de manière systématique

: Séquence d’étapes à effectuer (pour une sortie requise d’une certaine entrée donnée)

La sortie = la fin = résultat

Où ? programmation, math, vie quotidienne

Représentation en schéma de type logigramme

🡪Outil qui permet de visualiser les séquences de l algo

Une image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement

Lecture haut vers bas

Une image contenant texte, diagramme, Police, Plan

Description générée automatiquement

Programmation info

Machine = ordi

Considéré comme idiot qui se contente de faire ce qu’on lui dit de faire

Traite des infos binaires

/// Langages de programmation

Permet de communiquer avec une machine / traduise instruction

Html / CSS pas de langage de programmation mais présentation

Rubis / Python / PHP / JS ….

/// Ecrire un algo

Les variables :

Permettent de mémoriser des valeurs

Boite où déposer des valeurs, étiquetée

Ex : NbreJour, Moy, Date, Resultat\_01, Nom\_Famille

Tech nommage :

CamelCase : Mot collé mais MAJ à chaque début de mot (DateDuJour)

PascalCase : idem CamelCase ou (dateDuJour)

kebab-case : Date-Du-Jour (pas fou)

snake\_Case : Date\_Du\_Jour (python)

Variables : Affectation

Attribuer une valeur

Syntaxe : NomVariable 🡨 Valeur

Ici Valeur est évalué puis stocké dans NomVariable

Ex : N 🡨 7 // N prend comme valeur 7

Mess 🡨 « Bonjour » // variable mess va dire bonjour // pour les mots mettre des guillemets

A 🡨 B // A reçoit la valeur de B

Res 🡨 Nb1 + Nb2 // additione les contenue Nb1 et Nb2 et met le résultat

P 🡨 P +2 // ajoute 2 au contenu de P et met le résultat dans P

Typages

4 types : Entier : nbre sans virgule

Réel / flottant : nbre à virgule

Chaîne de caractères : « mot entre guillemet » ou « 3.14 »

Booléen : Vrai, Faux, true, False, 0, 1

!! Doit toujours être initialisé (affectation ou saisie)

Opérateurs

/ Sur variable Réelles :

+, -, x, /, a²

/ Sur variable entières (flottant) :

+, -, x sont ok

Cas particulier pour la division

On utilise ***div*** pour obtenir le quotient (// la division pour le quotient)

Et ***mod*** pour obtenir le reste (% pour le modulo)

Ex : 17 div 3 renvoie 5 et 17 mod 3 fournit 2

/ Sur les chaines de caractère :

Concaténation avec l’addition

Ex : A 🡨 « Salut, »

B 🡨 a + « tu vas bien ? »

B contiendra « Salut, tu vas bien ? »

Ex : Ne peut pas additionner : « 3 » + 4

« 6 » + « 4 » = 64 (pas d’espace)

« hello  » + « 3 » = « hello 3 » (espace car « hello » a un espace après)

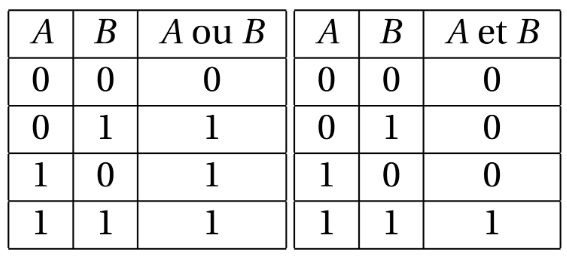
/ Sur les booléens

Opérateur logique :

Et, ou et Non

|  |
| --- |
| **X Y X ET Y X OU Y NON X NON Y** |
| **Vrai (1) Vrai (1) Vrai (1) Vrai (1) Faux (0) Faux (0)** |
| **Vrai (1) Faux (0) Faux (0) Vrai (1) Faux (0) Vrai (1)** |
| **Faux (0) Faux (0) Faux (0) Faux (0) Vrai (1) Vrai (1)** |

Table de vérité :



ET = \*

OU = +

Vrai = 1

Faux = 0

NON = inverse

NON( A + B) = A + B

Ex :

A=Vrai=1

B=Faux=0

C=Faux=0

D=Vrai=1

**A ET B OU C OU D ET D OU A ET C**

**=**

**1\*0 +0 + 1\*1 + 1\*0**

**(1\*0 =)0 + 0 + 1(=1\*1) + 0(=1\*0)**

**1**

Si 1 = Vrai

/// Lecture et Ecriture

Communique avec l’utilisateur

Lecture d’une donnée saisie au clavier

NomVariable 🡨saisir()

!!! Le programme se met en attente d’une entrée de l’utilisateur

Peut affichage saisir(« message »)

Ecriture d’un résultat / message

Afficher(« bonjour ») // Affiche la chaine de caractère bonjour

Afficher(bonjour) // Affiche le contenu de la variable bonjour

/// Langage python

# : commentaire

= : Affectation (🡨)

Input : saisir(« »)

Print : Afficher(« »)

Int() : transforme la chaine de caractère en chiffre entier

int("4") = 4

int(1) = 1

int(3.14) = 3

int(True) = 1

Int(False) = 0

int(b) = erreur

int(« 4.1 ») = erreur

Str() : transforme en chaine de caractère

Str = string

str ("4") = "4"

str(1) = "1"

str(3.14) = "3.14"

str(True) = "True"

str(False) = "False"

Float() : transforme en réelle, flotant

Float(1.2) = 1.2

Float(« 4.2 ») = 4.2

Float(« 4 ») = 4.0

Float(62) = 62.0

Float(true) = 1.0

Float(« hello ») = erreur

Bool() : transforme en booléen False quand 0, empty et none

Bool(« 4 ») = true

Bool(2) = true

Bool(0) = false

Bool(« hello ») = true

Bool(« ») = false

Bool(None) = false

Ex :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

 :

F (raccourci pour formater) pour pouvoir afficher les valeurs directement sans casser la chaine de caractère avec {} pour montrer les valeurs a afficher

Opérateurs de comparaison

Le résultat sera toujours un booléens (true or false)

= = : vérifie si 2 valeurs sont strictement identiques

! = : vérifie si 2 valeurs sont différentes

> : vérifie qu’une valeur est plus grande

< : vérifie qu’une valeur est plus petite

>= : supérieur ou égal

<= : inférieur ou égal

Structures de contrôle

Si : condition Alors

Si la condition est vérifiée on effectue un ou plusieurs traitements

* Vrai

Condition = opération logique avec des opérateurs logiques (ET/OU) et des opérateurs de comparaison

SON RESULTAT DONNE TOUJOURS UN BOOLEEN

Ex : Personne majeur ?

If Age >= 18 Alors :

Afficher (« vous êtes majeur »)

If Age < 18 Alors :

Afficher (« vous êtes mineur »)

Sinon : Inverse de la condition si

Ex : Personne majeur ?

If Age >= 18 Alors :

Afficher (« vous êtes majeur »)

Else :

Afficher (« vous êtes mineur »)

Round(résultat) = arrondi

Round(résultat, 2) = arrondi avec 2 chiffres après la virgule

Boucle Pour : (boucle comptée)

Pour variableDeComptage allant de début à fin Faire

🡪 Initialisée à début et va s’incrémenter de1 jusqu’à être = à fin

Instructions

FinPour

Ex : Pour i ***allant de*** 0 à 5, Faire

Afficher (1)

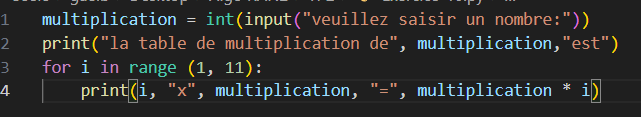
FinPour

For i ***in range*** (0 ,5) :

Print(i)

|  |  |  |
| --- | --- | --- |
| **Tours** | **i** | **I < 5** |
| **0** | **0** | **True** |
| **1** | **1** | **True** |
| **2** | **2** | **True** |
| **3** | **3** | **True** |
| **4** | **4** | **True** |
| **5** | **5** | **False** |

Ex :



Mot = « Hello »

For i in range (0,len(mot)) :

Print(mot[i])

H e l l o

0 1 2 3 4 : index, le dernier chiffre est le last index

Last index = longueur du mot -1

= len(mot) -1

Boucle While (tant que)

-Boucle conditionnelle

-fonctionne avec une condition de boucle

While condition : = on boucle si la condition est vérifiée

Instruction …..

Ex :

I = 0

While i < 10

Print(i)

I = i +1

Les Tableaux

Syntaxe : tab = [] # tableau vide

Tableau = liste

Tableau = array

Permet de stocker des valeurs

Ex : tab = [1, 2, 3]

Tab = [« a », « b », « c »]

Tab = [« a », 1, True, 4.4]

On peut dans un tableau : tab = [« a », « b », « c »]

Lire une valeur : tab[1]

Ajouter une valeur : tab = tab + [« d »] 🡪 [« a », « b », « c », « d »]

Tab.append(« d ») (méthode objet)

Modifier sa valeur : tab[2]= « e » 🡪 [« a », « b », « e », « d »]

Supprimer une valeur : del(tab[0]) 🡪 [« b », « e », »d »]

Notion de MVP : Minimum Viable Product

Cours liste :



Fonction python

Pourquoi ?

Pour réutiliser des traitements

Eviter les répétitions

Gagnez du temps

Réduire le code (plus lisible)

Quand ?

Il y a des traitements répétitifs

Trop de lignes

La fonction retourne une valeur alors que la procédure non

Ex :

Fonction Procédure

Def somme(a,b) : def somme(a,b)

Return a + b print(a + b)

Res = somme(5,4) somme(5,4)

Stock la valeur Affiche la valeur, on ne peut pas la réutiliser

Return sort la valeur de la fonction