1 Introduction à la programmation

A. Algorithme

Définition

Un **algorithme** est une liste ordonnée et logique d'instructions permettant de résoudre un problème.

Remarques :

- Il y a des algorithmes dans la vie courante : planter un clou, fermer à clé, visser, exécuter une recette de cuisine, monter un meuble préfabriqué ...
- Ils sont décrits en langage courant. Il peut y avoir plusieurs algorithmes différents pour effectuer une même tâche.

» **Exemple :** Planter un clou pour un droitier

- prendre le clou de la main gauche
- poser la pointe du clou à l'emplacement voulu
- prendre le marteau dans la main droite
- répéter « taper le clou avec le marteau » jusqu'à ce que le clou soit suffisamment enfoncé pour tenir seul
- retirer la main gauche
- répéter « taper le clou avec le marteau » jusqu'à ce que le clou soit complètement enfoncé

Remarques :

Un algorithme peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur. Ce sera l'objet du paragraphe B. Nous nous intéressons dans ce chapitre aux algorithmes qui seront utilisés par des machines.

Règle 1

Un **algorithme** se compose de trois grandes parties :
- les informations dont on a besoin au départ ;
- la succession d'instructions à appliquer ;
- la réponse que l'on obtient à l'arrivée.

» **Exemples**

Parties	Exemple 1	Exemple 2
Informations de départ	addition de deux nombres ; nombres A et B	recherche de la position de la lettre « a » dans un mot ; un mot
Succession d'instructions	calculer A+B	rechercher la position du caractère « a » dans le mot
Réponse à l'arrivée	la somme obtenue	la position trouvée

Remarques :

- La succession d'instructions n'est pas toujours détaillée explicitement au sein de l'algorithme, mais parfois dans une autre partie, à travers ce que l'on appelle des fonctions ou des procédures (voir le paragraphe 8).
- Cela permet de découper l'algorithme en plusieurs sous-algorithmes et de le rendre plus facile à comprendre.

Règle 2

Par souci de clarté, un algorithme doit éviter de comporter plusieurs fois la même série d'instructions.
Pour éviter cela on utilise, quand on le peut, les boucles (voir paragraphes 4 et 5) ou les fonctions et procédures.
On fait appel à la procédure au lieu de ré-écrire les mêmes instructions.
Cela permet aussi d'avoir des algorithmes et des programmes plus lisibles.

» **Exemple :** Pour tracer un carré de côté 3 cm

Algorithme :	Procédure « côté » : • Tracer un segment de 3 cm • Tourner de 90 ° à droite	
	Programme :	Programme plus efficace :
tracer un segment de 3 cm tourner de 90° à droite	« côté »	Répéter 4 fois « côté »
tracer un segment de 3 cm tourner de 90° à droite	« côté »	
tracer un segment de 3 cm tourner de 90° à droite	« côté »	
tracer un segment de 3 cm tourner de 90 ° à droite	« côté »	

↘ Entraîne-toi à Écrire un algorithme Niveau 1

- qui dessine un triangle équilatéral de côté 5cm.
- qui dessine un « Z » de « côté » 4 cm et de 5,7 cm de « diagonale ».

Correction

Algorithme :	Procédure « côté » : • tracer un segment de 5 cm • tourner de 120° à droite	
	Programme :	Programme plus efficace :
tracer un segment de 5 cm tourner de 120° à droite	« côté »	Répéter 3 fois « côté »
tracer un segment de 5 cm tourner de 120° à droite	« côté »	
tracer un segment de 5 cm tourner de 120° à droite	« côté »	

Correction

Algorithme :	Fonction « segment »(*longueur*) : tracer un segment de *longueur* cm
	Programme :
tracer un segment de 4 cm tourner de 135° à droite tracer un segment de 5,7cm tourner de 135° à gauche tracer un segment de 4 cm	segment (4) tourner de 135 ° à droite segment (5,7) tourner de 135 ° à gauche segment (4)

↘ Entraîne-toi à Comprendre un algorithme Niveau 1

Que fait l'algorithme suivant ?

Information de départ : le nombre x
Succession d'instructions :
donner à x la valeur x-3
donner à x la valeur $(x)^2$
Réponse à l'arrivée : la valeur de x

Correction

Il affiche la valeur de $(x-3)^2$ pour un nombre donné x.

B. Programmation

Remarques :

- Pour pouvoir communiquer avec les machines on utilise un langage de programmation (avec une syntaxe très précise). **Utilisateur** → algorithme → langage → ordinateur
- Pour nous aider, il existe des logiciels qui utilisent des langages plus simples (pseudo-codes) proches du langage courant. Le logiciel traduit ensuite en langage compréhensible par l'ordinateur sans que l'utilisateur ne le voie. **Utilisateur** → algorithme → pseudo-code → logiciel → langage → ordinateur

» **Exemple 1 :**

Logiciel de géométrie	Avec un tableur	Avec Scratch
Tracer [AB]	Calculer le nombre A+2	Avancer de 50 pas
Choix de l'action « tracer un segment » Clic sur le point A, Clic sur le point B.	Ecrire le nombre A en A1 Choisir la cellule A2 écrire : « =A1+2 »	Choisir la brique « avancer de 10 pas » Changer 10 par 50

Règle 3

En programmation les trois grandes parties d'un algorithme deviennent :

- les informations dont on a besoin au départ : les entrées ou la lecture
- la succession d'instructions à appliquer : le traitement
- la réponse que l'on obtient à l'arrivée : les sorties ou l'écriture

Remarques :

- Quand on dit « **lire** » ou « **écrire** » on se place du point de vue du programme ou de la machine.
- On **lit** les entrées : au clavier, à la souris (position ou clic) , à partir d'un fichier (image, texte, son), à partir d 'un capteur ... Cela arrête le programme et attend qu'une action soit réalisée par l'usager ou un autre programme afin d'obtenir une valeur.
- Le **traitement** n'est pas toujours détaillé dans le programme, mais dans une autre partie : les fonctions et procédures. Cela permet de découper un programme en plusieurs « sous programmes » et de le rendre ainsi plus facile à lire.
- Le traitement peut aussi comporter des entrées et des sorties.
- On « **écrit** » les sorties : affichage à l'écran, sur une imprimante, dans un fichier...
- Le langage algorithmique n'existe pas. C'est une convention d'écriture entre le livre, ou le professeur, et les élèves. Nous utiliserons dans la suite de ce livre un « langage algorithmique » couramment utilisé.

» **Exemple 2 :** Un algorithme pour ajouter 2 à un nombre A qui vaut 4.

Langage courant	Pseudo-code		Langage de programmation
	Scratch	Langage algorithmique	Python3

nombre de départ : A
attribuer à A la valeur 4
ajouter 2
le résultat est 6
		donner à A la valeur 4
		ajouter 2 à A
		écrire A
			`A=4`
			`A=A+2`
			`print(A)`

Entraîne-toi à Programmer un algorithme

Niveau 1

Programme un algorithme qui calcule $5(x + 3)$ pour un nombre donné x.

Correction

Langage courant	Pseudo-code		Langage de programmation
	Scratch	Langage algorithmique	Python3

choisir le nombre x
ajouter 3
multiplier le résultat par 5
le résultat est 5(x+3)
		lire le nombre x
		donner à x la valeur x+3
		donner à x la valeur 5*x
		écrire x
			`s = input('x= ')`
			`x=int(s)`
			`x=x+3`
			`x=5*x`
			`print(x)`

2 Les variables

A. Définition

Définition

Une **variable** est une information contenue dans une « boîte », que le programme va repérer par son nom. Pour avoir accès au contenu de la boîte, il suffit de la désigner par son nom. Le contenu de cette « boîte » dépend du type de variable. Il y a plusieurs types de variables:

- numérique : entier, réel — 1, -1 ou 2 sont entiers ; 0,5, π sont réels
- texte : caractère, chaîne — a ou h (caractères) « coucou » (chaîne)
- booléen — ne peut prendre que deux valeurs : vrai ou faux

» **Exemples :**

- la variable « A » peut représenter le nombre de frères et sœurs, c'est une variable numérique entière.
- la variable « Mot » peut représenter un nom, c'est une variable de type chaîne.
- la variable « VF » peut représenter un « vrai/faux », c'est une variable booléenne.

Remarques :

- Certains langages n'utilisent pas la déclaration de type. C'est le cas de Scratch et de Python.
- Attention ! Les nombres sont sans unité. L'unité dépend du logiciel utilisé.
- Pour les images, les dessins, on utilise souvent le pixel : le pixel (abréviation px) est le plus petit élément d'une image, il ne peut contenir qu'une seule couleur.

B. Affectation

Définition

Affecter une valeur à une variable, c'est donner une valeur à cette variable.

» **Exemple 1 :** Si j'affecte la valeur 3 à la variable A, la « boîte » A contiendra le nombre 3.
Ce qui signifie que j'ai 3 frères et sœurs.

Définition

On est tenté d'écrire A=3 , mais le signe « = » en programmation ne correspond pas au signe « = » en mathématiques. Il signifie qu'on attribue une valeur à une variable. C'est pourquoi on utilise une autre notation.

Écriture	Signification	Notation algorithmique
A=B	La « boîte » A reçoit la valeur qui était dans la « boîte » B	A ← B
B=A	La « boîte » B reçoit la valeur qui était dans la « boîte » A	B ← A
A=A+1	La « boîte » A reçoit sa valeur augmentée de 1	A ← A + 1

Dans la suite, nous utiliserons le symbole ← pour indiquer une affectation en « langage algorithmique ».

» **Exemple 2 :** L'instruction d'affectation consiste à « remplir » la « boîte » de la variable.

Langage algorithmique	Scratch	Python3
A ← 4 Mot ← «coucou »		`A=4` `Mot="coucou"`

» **Exemple 3 :** On peut aussi affecter à une variable la valeur d'une autre variable de même type.

Langage algorithmique	Scratch	Python3
A ← B Mot ← Mot2		`A=B` `Mot=Mot2`

» **Exemple 4 :** Entrée (ou saisie) de variable. Quand on lit une variable, c'est l'utilisateur qui donne la valeur.

Lange algorithmique	Scratch	Python3
lire un nombre A ← nombre		

C. Agir sur les variables

Définition

Pour agir sur les variables, on utilise des opérateurs qui dépendent du type de variables.

- numériques : + - * (multiplier) / (diviser) ^ (puissance)
- texte : & + (met bout à bout deux chaînes)
- logiques (booléen): « et » « ou » « non »

On utilise aussi de nombreuses fonctions prédéfinies (Voir l'annexe numérique des fonctions usuelles).

» **Exemple 1** : la syntaxe dépend des langages ou logiciels choisis.
ENT(nombre) renvoie la partie entière du nombre : ENT(2,5) = 2
MOD(entier1,entier2) renvoie le reste dans la division entière de entier1 par entier2 :
MOD(11,2)= 1

» **Exemple 2 :**

Langage algorithmique	Scratch	Python3
A ← 2(B+5) Mot ← Mot2 &« oui »		`A=2*(B+5)` `Mot=Mot2+"oui"`

Entraîne-toi à Affecter des valeurs à des variables — Niveau 1

Écris un programme qui calcule $(x+y)^2$ pour deux nombres donnés x et y.

Correction

Langage algorithmique	Scratch	Python3

variable x : réel
variable y : réel
lire x
lire y
x ← x+y
x ← x^2
écrire x
		`x=float(input("x="))`
		`y=float(input("y="))`
		`x=x+y`
		`x=x*x`
		`print("x=",x)`

Entraîne-toi à Comprendre les différents types de variables — Niveau 1

1) Que fait le programme suivant si on saisit la valeur « N » ?

```
variable x : caractère
lire x
écrire « vous avez saisi :  » +x
```

Correction

Il affiche « vous avez saisi : N»

2) Que fait le programme suivant si on saisit la valeur -5 ; -3,4 ; 0 .

```
variable x : nombre
variable test : booléen
lire x
test =  (x<0)
écrire test
```

Correction

Il lit le nombre x et affiche « vrai » si x<0 et « faux » sinon.

Pour -5 : vrai
Pour -3,4 : vrai
Pour 0 : faux

3 Les tests

Définition

Un **test** permet de choisir une action suivant une condition.

Structures d'un test

Si « condition est vraie » alors action1 fin de si
Si « condition est vraie » alors action1 Sinon action2 fin de si

» **Exemple 1 :** « Si j'ai faim alors je mange ».

» **Exemple 2 :** Division de B par A.

Langage algorithmique	Scratch	Python3

lire A et B
Si A n'est pas nul
alors
diviser B par A
écrire B
Sinon
écrire « impossible »
fin de si
		`A=int(input("A = "))`
		`B=int(input("B = "))`
		`if A!=0 :`
		`  B=B/A`
		`  print(" B= ",B)`
		`else :`
		`  print("impossible")`

Remarque : Qu'est-ce qu'une condition ?

En général une condition est une comparaison, elle est vraie ou fausse. La condition peut aussi être une variable de type booléen. On peut utiliser des opérateurs : « égal à » « différent de » « plus petit que » ....
Avec Python, il n'y a pas de « fin de... » mais on utilise l'**indentation** (retrait du texte qui permet de distinguer la partie de programme qui sera exécutée si la condition est réalisée).

» **Exemple 3 :** Dire si un nombre A est strictement négatif.

Langage algorithmique	Scratch	Python3

lire A
Si A<0 alors
écrire « A est strictement négatif »
fin de si
		`A=int(input("A = "))`
		`if A<0 :`
		`  print(" A est strictement négatif")`

Entraîne-toi à Utiliser un test « si... sinon... » Niveau 1

Écris un programme qui demande ton âge en années et te répond si tu es mineur ou majeur.

Correction

Langage algorithmique	Scratch	Python3

lire age
Si age<18 alors
écrire « Tu es mineur »
Sinon
écrire« Tu es majeur »
fin de si
		`Age=int(input("Age = "))`
		`if Age<18 :`
		`  print("Tu es mineur")`
		`else :`
		`  print("Tu es majeur")`

Entraîne-toi à Comprendre un programme utilisant un test Niveau 1

Dis ce que fait le programme suivant si N vaut 6 ; 5 ; -3 .

lire N
Si N est pair alors N ← N/2
Sinon N ← 3*N+1
fin de si
afficher N

Correction

Pour N=6 on affiche 3

Pour N=5 on affiche 16

Pour N=-3 on affiche -8

4 Les boucles « POUR » ou itération

Définition

Une **itération** sert à répéter une même action.

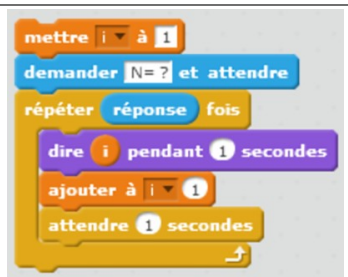
Remarque : On connaît le nombre de fois où l'action devra être répétée.
Une fois la répétition finie, le programme continue.
On doit décrire ce que l'on appelle un « **compteur de boucle** » :
- début : premier nombre
- fin : dernier nombre
- « pas » utilisé : de combien on augmente à chaque fois (ou 1 par défaut) .

» **Exemple 1 :** afficher 5 lignes de « coucou » (Avec Scratch on affiche 5 fois la ligne)

Langage algorithmique	Scratch	Python3
Répéter 5 fois : écrire « coucou » fin de répéter		`for i in range(5) :` `  print("coucou")`

» **Exemple 2 :** Afficher tous les entiers de 1 à N (donné) Avec Scratch, le lutin « compte ».

Langage algorithmique	Scratch	Python3

lire N entier
Répéter
pour i de 1 à N :
écrire i
fin de répéter
		`N=int(input("N= "))`
		`for i in range(1,N+1) :`
		`  print(i," ",end='')`

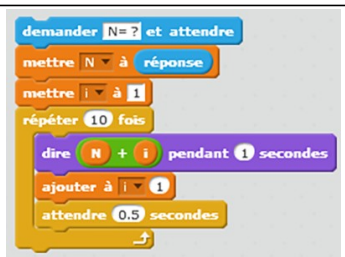
» **Exemple 3:** L'algorithme « Pour N allant de 1 à 10 pas 3 afficher N » donnera 1 puis 4 puis 7 puis 10.

Entraîne-toi à Utiliser une boucle « pour » Niveau 2

Écris un programme qui demande un nombre entier N et affiche tous les nombres de N+1 à N+10.

Correction Avec Scratch, le lutin « compte ».

Langage algorithmique	Scratch	Python3

lire N
Pour i allant de 1 à 10 (pas de 1)
écrire N+i
fin de pour
		`N=int(input("N= "))`
		`for i in range(1,11) :`
		`  print(N+i," ",end='')`

↘ Entraîne-toi à Comprendre un programme utilisant une boucle « pour » Niveau 2

Dis ce que fait le programme suivant si N vaut 2 ; 4 ; 1 ; 0 .

```
lire N entier
X =10
Répéter de 1 à N (pas de 1)
      X←    2*X
afficher X
```

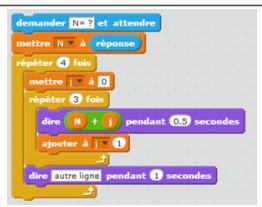
Correction

Pour N=2 on affiche 40 (2×2×10)
Pour N=4 on affiche 160 (2×2×2×2×10)
Pour N=1 on affiche 20
Pour N=0 on affiche 10
Il affiche le nombre $2^N \times 10$

```
N= 8
8   9   10
8   9   10
8   9   10
8   9   10
```

» **Exemple 4 :** Pour afficher l'écran ci-contre à partir d'un nombre N.

Langage algorithmique	Scratch	Python3

lire N entier
Répéter de i=1 à 4 :
  Répéter de j=0 à 2 :
  écrire N+j,« »
  fin de répéter
  sauter une ligne
fin de répéter
		N=int(input("N= "))
		for i in range(1,5) :
		  for j in range(0,3) :
		  print(N+j," ",end='')
		  print(" ")

5 Les boucles « TANT QUE »

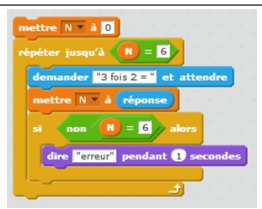
Définition

Une boucle « **Tant que** » sert à répéter une même action, jusqu'à ce qu'une condition se réalise.

Remarque : On ne sait pas à l'avance le nombre de fois que la boucle sera répétée.

» **Exemple 1 :** Demander « 3 fois 2 ». Lire la réponse N.
Tant que N est différent de 6 dire « erreur » .

Langage algorithmique	Scratch	Python3

afficher « 3 fois 2 = »
N ← 0
Tant que N ≠ 6 :
  lire N
  Si N ≠6
  Afficher « erreur »
  fin de si
fin de tant que
		N=0
		while N!=6 :
		  N=float(input("3 fois 2 = "))
		  if (N!=6) :
		  print("erreur")

On peut utiliser aussi cette boucle pour programmer un «faire ... jusqu'à ...» :

» **Exemple 2:** Lire un caractère A et l'afficher, jusqu'à ce que ce soit un « F ».

Langage algorithmique	Scratch	Python3

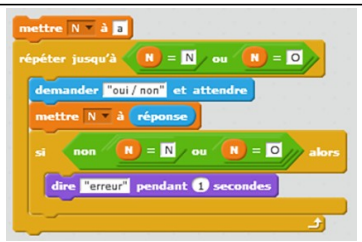
Répéter :
  lire A
  afficher A
jusqu'à ce que A soit « F »
		A=" "
		while A!="F" :
		  A=input("A= ")
		  print(A)

Entraîne-toi à Utiliser une boucle « tant que » Niveau 2

Écris un programme qui demande une réponse à « oui/non » et n'accepte que « O » ou « N » comme réponse valable. Sinon, il affiche « erreur ».

Correction

Langage algorithmique	Scratch	Python3

N ← «A»
Tant que (N n'est pas «O » ou «N ») :
afficher « oui / non »
lire N
Si (N n'est pas «O »ou «N ») :
afficher « erreur »
fin de si
fin du tant que
		N="a"
		while (N!="N") & (N!="O") :
		N=input(" Oui / Non = ")
		if (N!="N") & (N!="O") :
		print("erreur")

Entraîne-toi à Comprendre un programme utilisant une boucle « tant que » Niveau 2

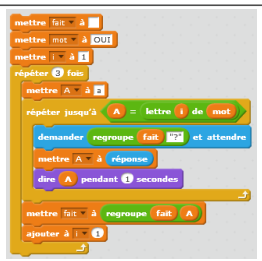
Dis ce que fait le programme suivant si N vaut 45 ; 27 ; 8 ; 10.

```
lire N entier
X←10
Tant que N>=X
    N ←  N-X
fin de tant que
écrire N
```

Pour N=45 on affiche 5
Pour N=27 on affiche 7
Pour N=8 on affiche 8
Pour N=10 on affiche 0
C'est le reste de la division euclidienne de N par 10

Correction

» **Exemple 3 :** Lire un caractère A et l'afficher, jusqu'à ce que ce soit le caractère, dans l'ordre, de « OUI »

Langage algorithmique	Scratch	Python3
Pour chaque caractère de « oui » : Répéter : lire A écrire A jusqu'à ce que A soit le bon caractère fin de pour		mot="OUI" fait="" for i in range(0,3): A=" " while A!=mot[i] : A=input(fait+"?= ") fait=fait+A

6 Scripts simultanés et événements déclencheurs

Cette partie ne concerne que le logiciel Scratch.

Avec Scratch, tu peux écrire un script pour un lutin.
Ce script est lancé par un **événement déclencheur.**
L'événement est à choisir dans le menu « événement ».
Pour chaque lutin et pour l'arrière-plan on peut définir un script différent.
Ces **scripts** seront **« simultanés »** s'ils sont lancés par le même événement.

» **Exemple 1 :** Par exemple, si les scripts débutent tous par alors le clic sur le drapeau vert va lancer les différents scripts.

Un **événement** lance le bloc qui lui est lié. C'est à dire la série d'instructions (briques) qui est collée dessous. On peut utiliser aussi différents événements à l'intérieur d'un même script.

Il suffit de séparer des blocs d'actions à exécuter en fonction de l'événement.

» **Exemple 2 :** Pour déplacer le lutin au clavier on utilise les touches « flèches » gauche et droite.

Événement

Un **événement** lance le bloc qui lui est lié. C'est à dire la série d'instructions (briques) qui est collée dessous. On peut utiliser aussi différents événements à l'intérieur d'un même script. Il suffit de séparer des blocs d'actions à exécuter en fonction de l'événement. Dans ce cas, le bloc qui interrompt est exécuté, puis le bloc interrompu reprend là où il s'était arrêté.

Message

Le programme en cours d'exécution peut aussi lancer un événement, en utilisant les **messages.** Cet événement ne dépend pas d'une action extérieure (clavier, souris,...) mais de l'exécution du programme.

7 Les tableaux, les listes

Définition

Ce sont des variables particulières. Elles sont utilisées pour stocker plusieurs variables de même type.

Un **tableau** est un ensemble de valeurs portant le même nom de variable et repérées par un nombre appelé **indice**.

Pour désigner un élément du tableau, on fait figurer le nom du tableau, suivi de l'indice de l'élément, entre crochets.

Attention, dans la plupart des langages de programmation, les indices des tableaux commencent à 0, et non à 1. C'est le cas de Python3. Dans un tableau nommé T, le 1er élément est alors T[0]. Pour Scratch les indices des listes commencent à 1.

» **Exemple 1 :** Lire 6 nombres et les ranger dans Tab.

Langage algorithmique	Scratch	Python3

Pour i de 0 à 5:
  lire un nombre
  Tab[i] ←valeur lue
fin de pour
		`Tab = []`
		`for i in range(6):`
		`  Tab.append(int(input("tab["+str(i)+"]= ")))`

↘ Entraîne-toi à Comprendre l'utilisation d'une liste Niveau 3

Dis ce que fait l'algorithme suivant :

variable x : tableau de 10 caractères
x← [a,b,c,d,e,f,g,h,i,j]
écrire x[2] +x[8]

Correction

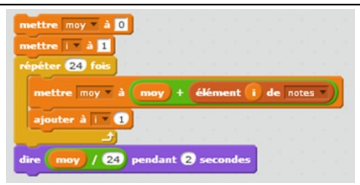
Il affiche « ci»

↘ Entraîne-toi à Utiliser une liste Niveau 3

Écris un algorithme qui calcule la moyenne des notes de 24 élèves, données dans un tableau « notes ».

Correction

Langage algorithmique	Scratch	Python3

moy ← 0
Pour i de 1 à 24 :
  moy ← moy +notes[i]
fin de pour
 	
moy← moy/24
écrire moy
		moy=0
		for i in range(24):
		  moy=moy+notes[i]
		print(notes)
		print("moyenne = ",moy/24)

Définition

La partie « tableaux multidimensionnels » qui suit ne concerne pas le logiciel Scratch.

Les Tableaux multidimensionnels sont utilisés pour faciliter la lecture (damier, image) ou tableau de tableaux : ce sont des tableaux dont chaque élément est lui-même un tableau.

» **Exemple 2 :** Pour utiliser les coordonnées (x;y) d'une série de points, on peut regrouper x et y en un seul tableau T. Voici un série de 6 points :
A (5 ; 2) B (8 ; -6) C (7 ; -1) D (9 ; 3) E (0 ; -4) F (6 ; 4)

	abscisse	ordonnée
A	5	2
B	8	-6
C	7	-1
D	9	3
E	0	-4
F	6	4

T[0][0] vaut 5 T[0][1] vaut 2
T[1][0] vaut 8 T[1][1] vaut -6
...
...
...
T[5][0] vaut 6 T[5][1] vaut 4

Le premier crochet correspond à la ligne et le deuxième à la colonne : T[ligne][colonne]

» **Exemple 3 :** pour une image de 100 x 200 pixels on peut utiliser un tableau pix (dessiné ci-dessous). pix contient tous les pixels rassemblés par ligne et colonne.
pix[3][5] correspond au 6e pixel de la 4e ligne de pix soit « A ».

N°	0	1	2	3	4	5	6	7	...	...	...								
0	e					c													
1				m															
2		s																	
3					n	A													
4		b					P												
....																			

Entraîne-toi à Comprendre l'utilisation d'un tableau

Niveau 3

Dis ce que fait l'algorithme suivant :

```
variable x : tableau [10, 2] de  caractères
x←[(  a,b,c,d,e,f,g,h,i,j),(k,k,k,k,k,k,k,k,k,k)]
x[5,1]←   x[5,0]
x[9,0]←   x[5,1]
écrire x[9,0]
```

Correction
après x[5,1]= x[5,0]
on a x=[(a,b,c,d,e,f,g,h,i,j),(k,k,k,k,k,f,k,k,k,k)]
après x[9,0]= x[5,1]
on a x=[(a,b,c,d,e,f,g,h,i,f),(k,k,k,k,k,f,k,k,k,k)]

Il affiche « f »

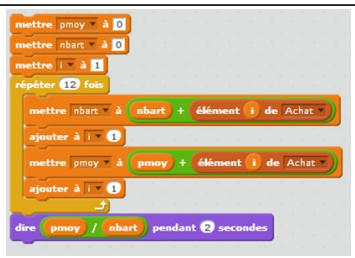
Entraîne-toi à Utiliser un tableau

Niveau 3

Écris un algorithme qui calcule le prix moyen d'un article.
On dispose d'un tableau « Achat », contenant pour chaque achat (12 en tout) :
- le nombre d'articles achetés
- le montant total payé.

Correction (Avec Scratch, on peut utiliser deux listes)

Langage algorithmique	Scratch	Python3

pmoy ← 0
nbart ← 0
 	
Pour i de 1 à 12 :
  pmoy←pmoy +Achat[i][1]
  nbart←nbart +Achat[i][0]
fin de pour
 	
pmoy← pmoy/nbart
afficher pmoy
		pmoy=0
		nbart=0
		for i in range(0,12,1):
		  pmoy=pmoy+Achat[1][i]
		  nbart=nbart+Achat[0][i]
		print(Achat)
		print("moyenne = ",pmoy/nbart)

8 Les fonctions et procédures

Définition

Les **fonctions** et **procédures** sont des « morceaux de programme » que l'on peut appeler en leur indiquant des paramètres.
On réutilise souvent la même partie d'un programme. Au lieu de ré-écrire cette partie, on en fait une fonction ou une procédure.

» **Exemple 1 :**
Proc () est une procédure nommée « Proc » sans paramètre.
Fonc (par1,par2) est une fonction nommée « Fonc » avec deux paramètres.
int FONC (par1) est une fonction nommée « FONC » avec un paramètre et qui renvoie une valeur entière.

» **Exemple 2 :** Pour tracer un carré de côté 5 cm on répète 4 fois : tracer un segment de 5 cm puis tourner de 90° à droite.

Au lieu de :
tracer un segment de 5 cm
tourner de 90° à droite
tracer un segment de 5 cm
tourner de 90° à droite
tracer un segment de 5 cm
tourner de 90° à droite
tracer un segment de 5 cm
tourner de 90° à droite

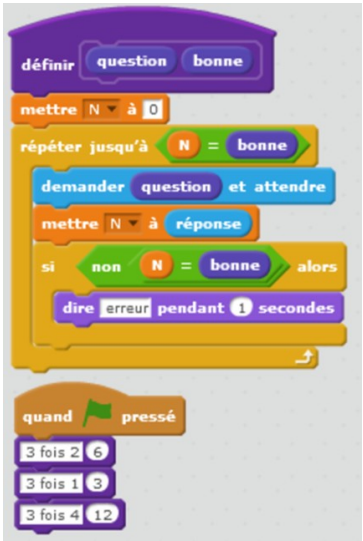
Cote () :
  tracer un segment de 5 cm
  tourner de 90° à droite

Programme :
répéter 4 fois Cote()
Ici, on a une procédure sans paramètre.

» Exemple 3 :

Nous avions vu le programme : Demander « 3 fois 2 » ». Lire la réponse N. Tant que N est différent de 6 dire « erreur ».
Ce petit programme peut devenir une fonction « Dem (texte,valeur) » et on peut l'appeler avec des paramètres différents. Avec Scratch, la fonction ne peut pas « retourner » une valeur, on doit utiliser une variable.

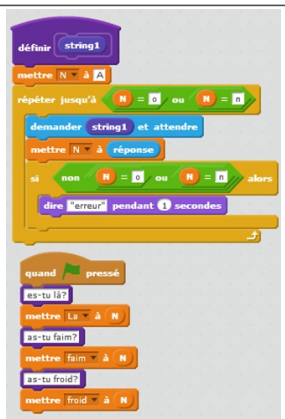
Langage algorithmique	Scratch	Python3

Dem (texte, valeur) :
écrire texte
N ← 0
Tant que N ≠ valeur :
lire N
Si N ≠valeur
écrire« erreur »
fin de si
fin de tant que
 	
Programme :
Dem («3 fois 2»,6)
Dem («3 fois 1»,3)
Dem («2 fois 4»,8)
		def Dem(texte,valeur):
		N=0
		while N!=valeur :
		N=float(input(texte))
		if (N!=valeur) :
		print("erreur")
		return N
		
		Dem("3 fois 2 ",6)
		Dem("3 fois 1 ",3)
		Dem("2 fois 4 ",8)

» Exemple 4 :

Nous avions vu le programme : Demande une réponse à « oui/non » et n'accepte que « O » ou « N » comme réponse valable. Sinon, affiche « erreur ».
Ce petit programme peut devenir une fonction qui renvoie la réponse à « oui/non ».
C'est très utile lors de la saisie de questionnaire.
Avec Scratch, la fonction ne peut pas « retourner » une valeur, on doit utiliser une variable.

Langage algorithmique	Scratch	Python3

Rep :
N ← «A»
Tant que (N n'est pas
«O »ou «N ») :
écrire « oui / non »
lire N
Si (N n'est pas
«O »ou «N ») :
écrire « erreur »
fin de si
fin de tant que
renvoyer N
 	
Programme :
écrire («es-tu là?»)
la← Rep
écrire(«as-tu faim?»)
faim← Rep
écrire («as-tu froid?»)
froid← Rep
		def Rep():
		N="a"
		while (N!="N") & (N!="O") :
		N=(input("oui / non = "))
		if (N!="N") & (N!="O") : print("erreur")
		return N
		
		print("es-tu là ?")
		la=Rep()
		print("as-tu faim ?")
		faim=Rep()
		print("as-tu froid ?")
		froid=Rep()

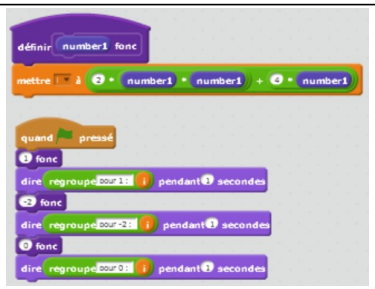
↘ Entraîne-toi à Utiliser une fonction qui renvoie une valeur

Niveau 3

Écris une fonction qui utilise le nombre X en paramètre et renvoie le résultat :
$2X^2 + 4\,X$.

Correction Avec Scratch, la fonction ne peut pas « retourner » une valeur, on doit utiliser une variable.

Langage algorithmique	Scratch	Python3

Fonc(X) :
C ← 2*X² + 4*X
renvoyer C
 	
Programme :
écrire(Fonc(1))
écrire(Fonc(-2))
écrire(Fonc(0))
		`def Fonc(X):`
		`  C=2*X*X+4*X`
		`  return C`
		
		`print("pour 1 :",Fonc(1))`
		`print("pour -2 :",Fonc(-2))`
		`print("pour 0 :",Fonc(0))`

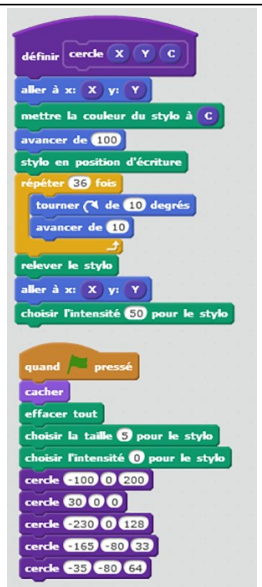
↘ Entraîne-toi à Utiliser une procédure

Niveau 3

Écris une fonction qui utilise les nombres X, Y et C en paramètres et affiche le cercle de centre (X;Y) et de rayon 100 pixels en couleur C. L'utiliser pour tracer les anneaux olympiques

Correction

Langage algorithmique	Scratch	Python3

Cercle(X,Y, C) :
aller en position (X;Y)
choisir la couleur C
tracer le cercle de 100 px
 	
Programme :
Cercle(-100,0,noir)
Cercle(30,0,rouge)
Cercle(-230,0,bleu)
Cercle(-165,-80,jaune)
Cercle(-35,-80,vert)
		`from tkinter import *`
		
		`def cercle(x,y,c):`
		`  canvas.create_oval(x-50, y-50, x+50, y+50, outline=c, width=5)`
		
		`# Ouverture de fenêtre`
		`fenetre = Tk()`
		`label = Label(fenetre, text="Anneaux Olympiques")`
		`label.pack()`
		`canvas = Canvas(fenetre, width=400, height=400, background='white')`
		`canvas.pack()`
		`# Dessin`
		`cercle(200, 130, 'black')`
		`cercle(330, 130, 'red')`
		`cercle(70, 130, 'blue')`
		`cercle(135, 180, 'yellow')`
		`cercle(265, 180, 'green')`
		
		`fenetre.mainloop()`