1

TP : Algorithmique - Initiation à python

1. Structures de base en langage naturel

1) Variables et affectations

Les variables en algorithmique

- Les variables algorithmiques peuvent servir à stocker des données de différents types, mais nous nous contenterons ici d'utiliser des variables du type NOMBRE.
- La valeur d'une variable peut changer au fil des instructions de l'algorithme.
- Les opérations sur les variables s'effectuent ligne après ligne et les unes après les autres.
- Quand une ligne du type « mavariable ← un calcul », on effectue d'abord le calcul et on stocke ensuite le résultat dans mavariable.

► Activité n°1

On considère l'algorithme suivant :

```
Variables: x,y,z
1: DEBUT_ALGORITHME
2:     x ← 2
3:     y ← 3
4:     z ← x+y
5: FIN_ALGORITHME
```

Après exécution de l'algorithme : la variable x contient la valeur : ☐ ;

la variable y contient la valeur : ☐ ; la variable z contient la valeur : ☐

► Activité n°2

On considère l'algorithme suivant :

```
Variables: x
1: DEBUT_ALGORITHME
2:     x ← 2
3:     x ← x+1
4: FIN_ALGORITHME
```

Après exécution de l'algorithme, la variable x contient la valeur : ☐

► Activité n°3

Ajoutons la ligne « x ← 4*x » à la fin du code précédent. Ce qui donne :

```
Variables: x
1: DEBUT_ALGORITHME
2:     x ← 2
3:     x ← x+1
4:     x ← 4*x
5: FIN_ALGORITHME
```

Après exécution de l'algorithme, la variable x contient la valeur : ☐

► Activité n°4

On considère l'algorithme suivant :

```
Variables: A,B,C
1: DEBUT_ALGORITHME
2:     A ← 5
3:     B ← 3
4:     C ← A+B
5:     B ← B+A
6:     A ← C
7: FIN_ALGORITHME
```

Après exécution de l'algorithme : la variable A contient la valeur : ☐ ;

la variable B contient la valeur : ☐

la variable C contient la valeur : ☐

► Activité n°5

On considère l'algorithme suivant :

```
Variables: x,y,z
1: DEBUT_ALGORITHME
2:     LIRE x
3:     y ← x-2
4:     z ← -3*y-4
5:     AFFICHER z
6: FIN_ALGORITHME
```

On cherche maintenant à obtenir un algorithme équivalent sans utiliser la variable y. Compléter la ligne 3 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
Variables: x,z
1: DEBUT_ALGORITHME
2:     LIRE x
3:     z ← ..................
4:     AFFICHER z
5: FIN_ALGORITHME
```

2) Instructions conditionnelles

SI...ALORS...SINON

Comme nous l'avons vu ci-dessus, un algorithme permet d'exécuter une liste d'instructions les unes à la suite des autres. Mais on peut aussi demander à un algorithme de n'exécuter des instructions que si une certaine condition est remplie. Cela se fait grâce à au bloc d'instructions SI...ALORS :

```
SI...ALORS
  ...
  FIN_SI
```

Il est aussi possible d'indiquer en plus à l'algorithme de traiter le cas où la condition n'est pas vérifiée. On obtient alors la structure suivante :

```
SI...ALORS
  ...
  FIN_SI
SINON
  ...
  FIN_SINON
```

► Activité n°6

On cherche à créer un algorithme qui demande un nombre à l'utilisateur et qui affiche la racine carrée de ce nombre s'il est positif. Compléter la ligne 3 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
Variables: x,racine
1: DEBUT_ALGORITHME
2:     LIRE x
3:     SI (.........) ALORS
4:         racine ← √x
5:         AFFICHER racine
6:         FIN_SI
7: FIN_ALGORITHME
```

► Activité n°7

On cherche à créer un algorithme qui demande à l'utilisateur d'entrer deux nombres (stockés dans les variables x et y) et qui affiche le plus grand des deux. Compléter les ligne 5 et 8 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
Variables: x,y
1: DEBUT_ALGORITHME
2:     LIRE x
3:     LIRE y
4:     SI (x>y) ALORS
5:         AFFICHER .......
6:         FIN_SI
7:     SINON
8:         AFFICHER ......
9:         FIN_SINON
10: FIN_ALGORITHME
```

► Activité n°8

On considère l'algorithme suivant :

```
Variables: A,B
DEBUT_ALGORITHME
    A ← 1
    B ← 3
    SI (A>0) ALORS
        A ← A+1
        FIN_SI
    SI (B>4) ALORS
        B ← B-1
        FIN_SI
FIN_ALGORITHME
```

Après exécution de l'algorithme :

- La variable A contient la valeur : ☐
- La variable B contient la valeur : ☐

► Activité n°9

On cherche à concevoir un algorithme correspondant au problème suivant :

- on demande à l'utilisateur d'entrer un nombre (représenté par la variable x)
- si le nombre entré est différent de 1, l'algorithme doit stocker dans une variable y la valeur de 1/(x-1) et afficher la valeur de y (note : la condition x différent de 1 s'exprime avec le code x!=1). On ne demande pas de traiter le cas contraire.

Compléter l'algorithme ci-dessous pour qu'il réponde au problème.

```
Variables: x,y
DEBUT_ALGORITHME
    LIRE ......
    SI (........) ALORS
        ....... ← .......
        AFFICHER .......
        FIN_SI
FIN_ALGORITHME
```

3) Boucles

Boucles POUR...DE...A

- Les boucles permettent de répéter des instructions autant de fois que l'on souhaite.
- Lorsqu'on connait par avance le nombre de fois que l'on veut répéter les instructions, on utilise une boucle du type POUR...DE...A dont la structure est la suivante :

```
POUR...ALLANT_DE...A...
    ...
FIN_POUR
```

- Exemple : l'algorithme ci-dessous permet d'afficher la racine carrée de tous les entiers de 1 jusqu'à 50.

```
Variables: n et racine
DEBUT_ALGORITHME
    POUR n ALLANT_DE 1 A 50
        racine ← √n
        AFFICHER racine
        FIN_POUR
FIN_ALGORITHME
```

La variable n est appelée « compteur de la boucle ».

- Remarques :
  - La variable servant de compteur pour la boucle doit être du type NOMBRE et doit être déclarée préalablement (comme toutes les variables).
  - Par défaut, cette variable est automatiquement augmentée de 1 à chaque fois.
  - On peut utiliser la valeur du compteur pour faire des calculs à l'intérieur de la boucle, mais les instructions comprises entre POUR et FIN_POUR ne doivent en aucun cas modifier la valeur de la variable qui sert de compteur.

► Activité n°10

On cherche à concevoir un algorithme qui affiche, grâce à une boucle POUR...DE...A, les résultats des calculs suivants : 8*1 ; 8*2 ; 8*3 ; 8*4 ; ... jusqu'à 8*10 .

La variable n sert de compteur à la boucle et la variable produit sert à stocker et afficher les résultats. Compléter les lignes 2 et 3 dans l'algorithme ci-dessous pour qu'il réponde au problème :

```
Variables: n,produit
DEBUT_ALGORITHME
    POUR n ALLANT_DE .... A .....
        produit ← .......
        AFFICHER produit
        FIN_POUR
FIN_ALGORITHME
```

► Activité n°11

On considère l'algorithme suivant :

```
Variables: n,somme
1: DEBUT_ALGORITHME
2:     somme ← 0
3:     POUR n ALLANT_DE 1 A 100
4:         somme ← somme+n
5:         FIN_POUR
6:     AFFICHER somme
7: FIN_ALGORITHME
```

Compléter les phrases suivantes :

- Après exécution de la ligne 2, la variable somme contient la valeur : ☐
- Lorsque le compteur n de la boucle vaut 1 et après exécution du calcul ligne 4, la variable somme vaut : ☐
- Lorsque le compteur n de la boucle vaut 2 et après exécution du calcul ligne 4, la variable somme vaut : ☐

Que permet de calculer cet algorithme ?

☐

► Activité n°12

Compléter les lignes 3 et 4 de l'algorithme ci-dessous pour qu'il permette de calculer la somme $5^2 + 6^2 + 7^2 + \cdots + 24^2 + 25^2$.

```
Variables: n, somme
1: DEBUT_ALGORITHME
2:     somme ← 0
3:     POUR n ALLANT_DE .... A ......
4:         somme ← somme+......
5:         FIN_POUR
6:     AFFICHER somme
7: FIN_ALGORITHME
```

Boucles TANT QUE...

- Il n'est pas toujours possible de connaitre par avance le nombre de répétitions nécessaires à un calcul. Dans ce cas là, il est possible d'avoir recours à la structure TANT QUE... qui se présente de la façon suivante :

```
TANT_QUE...FAIRE
  ...
  FIN_TANT_QUE
```

Cette structure de boucle permet de répéter une série d'instructions (comprises entre TANT_QUE...FAIRE et FIN_TANT_QUE) tant qu'*une certaine condition est vérifiée*.

- Exemple : Comment savoir ce qu'il reste si on enlève 25 autant de fois que l'on peut au nombre 583 ? Pour cela on utilise une variable n, qui contient 583 au début, à laquelle on enlève 25 tant que c'est possible, c'est à dire tant que n est supérieur ou égal à 25.

```
Variables: n
1: DEBUT_ALGORITHME
2:     n ← 583
3:     TANT_QUE (n>=25) FAIRE
4:         n ← n-25
5:         FIN_TANT_QUE
6:     AFFICHER n
7: FIN_ALGORITHME
```

- Remarques :
  - Si la condition du TANT QUE... est fausse dès le début, les instructions entre TANT_QUE...FAIRE et FIN_TANT_QUE ne sont jamais exécutées (la structure TANT QUE ne sert alors strictement à rien).
  - Il est indispensable de s'assurer que la condition du TANT QUE... finisse par être vérifiée (le code entre TANT_QUE...FAIRE et FIN_TANT_QUE doit rendre vraie la condition tôt ou tard), sans quoi l'algorithme ne pourra pas fonctionner.

► Activité n°13

On cherche à connaître le plus petit entier N tel que 2^N soit supérieur ou égal à 10000. Pour résoudre ce problème de façon algorithmique :

- On utilise une variable N à laquelle on donne au début la valeur 1.
- On augmente de 1 la valeur de N tant que 2^N n'est pas supérieur ou égal à 10000.

Une structure TANT QUE est particulièrement adaptée à ce genre de problème car on ne sait pas a priori combien de calculs seront nécessaires.

Compléter les lignes 3 et 4 de l'algorithme ci-dessous pour qu'il réponde au problème :

```
Variables: N
1: DEBUT_ALGORITHME
2:     N ← 1
3:     TANT_QUE (2^N.......) FAIRE
4:         N ← .......
5:         FIN_TANT_QUE
6:     AFFICHER N
7: FIN_ALGORITHME
```

► Activité n°14

Un individu a emprunté à un ami une somme de 2500 euros (prêt sans intérêts). Pour rembourser son ami, il prévoit de lui remettre 110 euros par mois. Mais comme cela ne correspond pas à un nombre pile de mois, il se demande quel sera le montant à rembourser le dernier mois.

Compléter la ligne 3 dans l'algorithme ci-dessous pour qu'il réponde au problème :

```
Variables: montant
1: DEBUT_ALGORITHME
2:     montant ← 2500
3:     TANT_QUE (.............) FAIRE
4:         montant ← montant-110
5:         FIN_TANT_QUE
6:     AFFICHER montant
7: FIN_ALGORITHME
```

► Activité n°15

On considère le problème suivant :

- On lance une balle d'une hauteur initiale de 300 cm.
- On suppose qu'à chaque rebond, la balle perd 10% de sa hauteur (la hauteur est donc multipliée par 0.9 à chaque rebond).
- On cherche à savoir le nombre de rebonds nécessaire pour que la hauteur de la balle soit inférieure ou égale à 10 cm.

Compléter les lignes 4 et 6 de l'algorithme ci-dessous pour qu'il réponde au problème.

```
Variables: nombre,hauteur
1: DEBUT_ALGORITHME
2:     nombre_rebonds ← 0
3:     hauteur ← 300
4:     TANT_QUE (hauteur......) FAIRE
5:         nombre_rebonds ← nombre_rebonds+1
6:         hauteur ← ...........
7:         FIN_TANT_QUE
8:     AFFICHER nombre_rebonds
9: FIN_ALGORITHME
```

4) Fonctions

Les fonctions en algorithmique

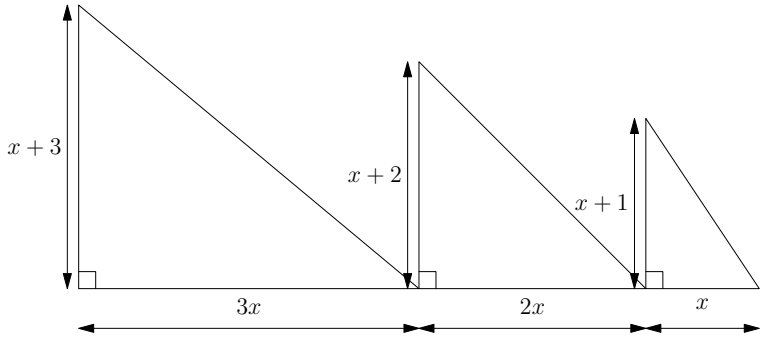
Pour éviter de répéter certaines instructions, on peut utiliser une fonction, c'est à dire une bloc d'instructions qui ne sera exécuté que quand on l'appelle. Une fonction dépend en général d'un ou plusieurs paramètres et retourne une valeur dépendant de ces paramètres.

```
FONCTION mafonction(paramètres...)
DEBUT_FONCTION
  ...
  RETOURNER...
FIN_FONCTION
```

Utiliser une fonction en algorithmique n'a d'intérêt que si on l'appelle plusieurs fois.

► Activité n°16

On cherche à élaborer un algorithme qui donne en fonction de x la somme des hypoténuses des trois triangles rectangles de la figure ci-dessous :



Pour cela, on va utiliser une fonction `hypo` qui renvoie l'hypoténuse d'un triangle rectangle dont les côtés de l'angle droit sont `a` et `b`.

Compléter les lignes 3 et 7 pour que l'algorithme réponde au problème posé :

```
Variables: x
1: FONCTION hypo(a,b)
2: DEBUT_FONCTION
3:     Retourner .................
4: FIN_FONCTION
5: DEBUT_ALGORITHME
6:     Lire x
7:     AFFICHER hypo(3*x,x+3)+hypo(2*x,x+2)+hypo(.....,.....)
8: FIN_ALGORITHME
```

2. Initiation à python

1) Introduction

Langage informatique

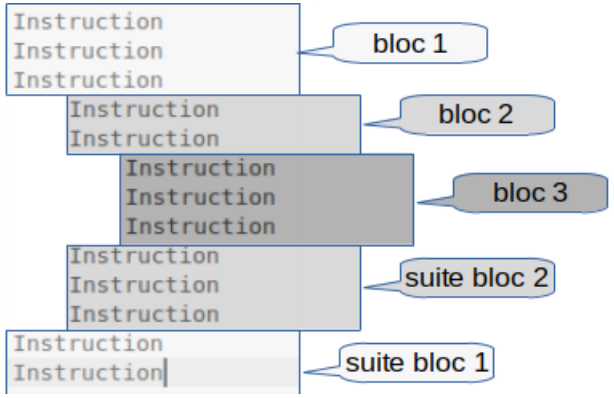
Un langage informatique permet de traduire un algorithme en une série d'instructions compréhensibles et exécutables par un ordinateur. Pour cela, les instructions écrites dans le langage informatique doivent respecter des spécificités et notamment une syntaxe plus précise et plus compliquée que celle d'un langage naturel.

Quelques principes de base à propos du langage python

- Par défaut, le langage python n'inclut pas bon nombre de fonctions mathématiques. C'est pour cela que certains scripts python commenceront par l'instruction suivante qui indique à python de charger les fonctions mathématiques standard :

```
from math import*
```

- Les débuts et fins de bloc d'instructions se marquent en langage python par un décalage horizontal (appelé indentation) et pas par des instructions particulières. Il faut donc faire très attention en python à ce que les instructions d'un même bloc soient écrites avec le même décalage horizontal par rapport à ce qui précède et suit ce bloc (on dit que ces instructions doivent avoir la même indentation).



2) Équivalences entre langage naturel et python

Affectation d'une valeur à une variable x

Langage naturel	Équivalent en python
x ← valeur	x = valeur

Lire la valeur d'une variable x entrée au clavier

Si x est un entier (et si l'algorithme ne manipule que des entiers) :

Langage naturel	Équivalent en python
LIRE x	x = int(input("x?"))

Si x est un réel :

Langage naturel	Équivalent en python
LIRE x	x = float(input("x?"))

Si on a besoin d'entrer une valeur résultant d'un calcul mathématique (fraction, racine carrée...) :

Langage naturel	Équivalent en python
LIRE x	from math import* x = eval(input("x?"))

Afficher la valeur d'une variable x

Langage naturel	Équivalent en python
AFFICHER x	print(x)

Afficher un message

Langage naturel	Équivalent en python
AFFICHER "Message"	print("message")

Les opérations mathématiques de base

Langage naturel	Équivalent en python
resultat ← a+b resultat ← a-b resultat ← $a \times b$ resultat ← $\frac{a}{b}$ resultat ← a^2 resultat ← $\sqrt{a}$	from math import* resultat=a+b resultat=a-b resultat=a*b resultat=a/b resultat=a**2 resultat=sqrt(a)

Pour a et b entiers	Équivalent en python
reste de la division euclidienne de a par b	a%b
quotient de la division euclidienne de a par b	a//b

► **Activité n°17**

Compléter le code du script python ci-dessous pour qu'il corresponde à l'algorithme en langage naturel.

Langage naturel	Équivalent en python
LIRE distance LIRE temps vitesse ← $\frac{\text{distance}}{\text{temps}}$ AFFICHER vitesse	distance=float(input("distance?")) temps=..... vitesse=..... print(vitesse)

► **Activité n°18**

Compléter le script python ci-dessous pour qu'il affiche la valeur de la diagonale d'un rectangle après avoir entré sa largeur et sa longueur :

Script python
from math import* largeur=float(input("largeur?")) longueur=..... diagonale=..... print(.....)

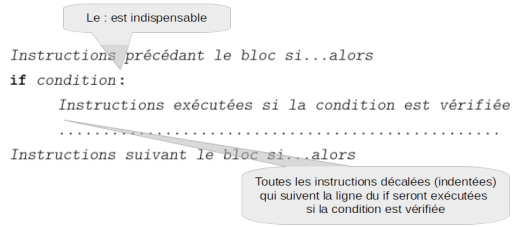
Instructions conditionnelles

• Avec un SI...ALORS

Langage naturel

Instructions précédant le bloc si...alors
SI *condition* ALORS
  Instructions exécutées si la condition est vérifiée
  .....
  FIN_SI
Instructions suivant le bloc si...alors

Équivalent en python



► **Exemple :**

Langage naturel	Équivalent en python

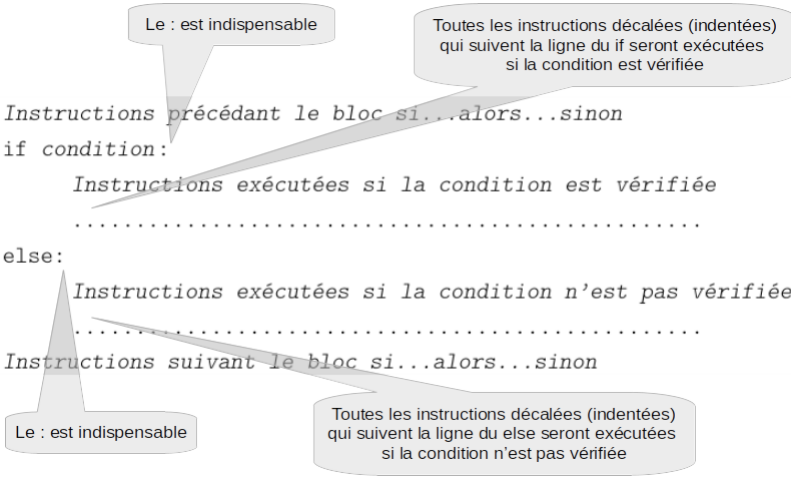
LIRE x
SI x>0 ALORS
  AFFICHER "x est strictement positif"
  FIN_SI
AFFICHER "son opposé est"
AFFICHER -x
	x=float(input("x?"))
	if x>0:
	  print("x est strictement positif")
	print("son opposé est",-x)

• Avec un SI...ALORS...SINON

Langage naturel

Instructions précédant le bloc si...alors...sinon
SI *condition* ALORS
  Instructions exécutées si la condition est vérifiée
  .....
SINON
  Instructions exécutées si la condition n'est pas vérifiée
  .....
  FIN_SI
Instructions suivant le bloc si...alors...sinon

Équivalent en python



• Syntaxe python pour les conditions

Situation	Syntaxe python correspondante
Si a égal à b (a et b entiers)	if a==b:
Si a est différent de b (a et b entiers)	if a!=b:
Si a est strictement supérieur à b	if a>b:
Si a est strictement inférieur à b	if a<b:
Si a est inférieur ou égal à b	if a<=b:
Si a est supérieur ou égal à b	if a>=b:

Cas particulier de la comparaison de deux réels de type float (qui ne sont pas représentés de façon exacte en informatique) :

Situation mathématique	Syntaxe pour python (>=3.5)
Si a égal à b (a et b flottants)	from math import*
Si a est différent de b (a et b flottants)	if isclose(a,b):
	if not(isclose(a,b)):

► Activité n°19

Compléter le code du script python ci-dessous pour qu'il corresponde à l'algorithme en langage naturel.

Langage naturel	Équivalent en python
LIRE age	age=int(input("age?"))
SI age>=18 ALORS	.....
AFFICHER "majeur"	.....
FIN_SI	

► Activité n°20

Parmi les quatre scripts python ci-dessous, déterminer le seul valide indiquant le plus grand de deux entiers.

Script 1	Script 2

a=int(input("a?"))
b=int(input("b?"))
if a>b
  print("le plus grand est a")
else:
  print("le plus grand est b")
	a=int(input("a?"))
	b=int(input("b?"))
	if a>b:
	  print("le plus grand est b")
	else:
	  print("le plus grand est a")

Script 3	**Script 4**

a=int(input("a?"))
b=int(input("b?"))
if a>b:
  print("le plus grand est a")
else:
  print("le plus grand est b")
	a=int(input("a?"))
	b=int(input("b?"))
	if a>b:
	  print("le plus grand est a")
	else:
	print("le plus grand est b")

• Opérateurs ET/OU/CONTRAIRE dans les conditions

Situation	Syntaxe python correspondante
Si *condition1* OU *condition2*	if *condition1* or *condition2*:
Si *condition1* ET *condition2*	if *condition1* and *condition2*:
Si CONTRAIRE(*condition*)	if not(*condition*):

► **Remarque :** or, and et not doivent être en minuscules

► **Exemple :**

Langage naturel	Équivalent en python

LIRE x
SI x>0 ET X<1 ALORS
  AFFICHER "x est dans]0;1["
  FIN_SI
	x=float(input("x?"))
	if x>0 and x<1:
	  print("x est dans]0;1[")

► Activité n°21

On considère l'expression $A(x) = \dfrac{1}{(x-1)\sqrt{x}}$ où x est un réel.

Compléter le code du script python ci-dessous pour qu'il corresponde à l'algorithme en langage naturel.

Langage naturel	Équivalent en python

LIRE x
SI x>0 ET x≠1 ALORS
  AFFICHER "A(x) existe"
SINON
  AFFICHER "A(x) n'existe pas"
  FIN_SI
	from math import*
	x=.....
	if.....
	.....
	else:
	.....

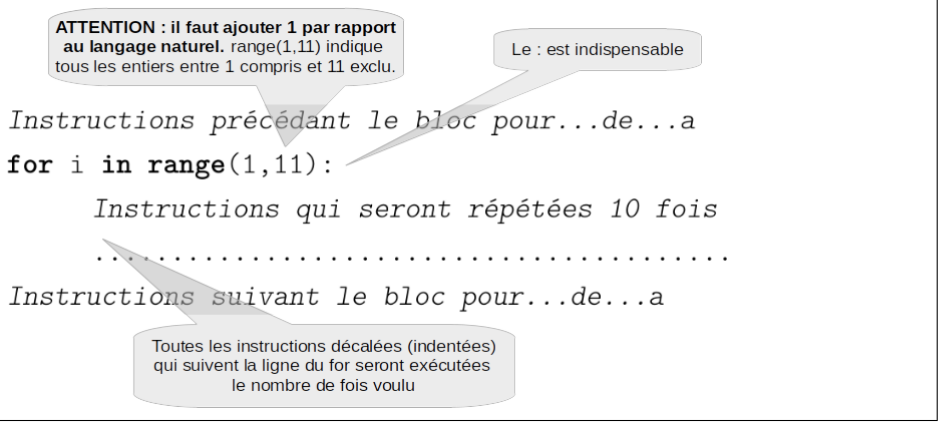
Boucles POUR...DE...A

• Rappel : ces boucles permettent de répéter des instructions un certain nombre (connu par avance) de fois.

Langage naturel

Instructions précédant le bloc pour...de...a
POUR i ALLANT_DE 1 A 10
  Instructions qui seront répétées 10 fois
  .....
  FIN_POUR
Instructions suivant le bloc pour...de...a

Équivalent en python



► **Exemple :**

Langage naturel	Équivalent en python

POUR i ALLANT_DE 1 A 10
  AFFICHER 7×i
  FIN_POUR
AFFICHER "script terminé"
	for i in range(1,11):
	  print(7*i)
	print("script terminé")

► **Activité n°22**

Compléter le code du script python ci-dessous pour qu'il corresponde à l'algorithme en langage naturel dont le but est de calculer la somme $1+2+3+\cdots+99+100$.

Langage naturel	Équivalent en python

somme ← 0
POUR i ALLANT_DE 1 A 100
  somme ← somme+i
  FIN_POUR
AFFICHER somme
	somme=0
	for i .....
	  somme=.....
	print(somme)

► **Activité n°23**

Parmi les quatre scripts python ci-dessous, déterminer le seul valide qui permet de calculer la somme $\sqrt{1}+\sqrt{2}+\sqrt{3}+\cdots+\sqrt{49}+\sqrt{50}$.

Script 1	Script 2

from math import*
somme=0
for i in range(1,51)
  somme=somme+sqrt(i)
print(somme)
	from math import*
	somme=0
	for i in range(1,51):
	  somme=somme+sqrt(i)
	print(somme)

Script 3	Script 4
from math import* somme=0 for i in range(1,50): somme=somme+sqrt(i) print(somme)	from math import* somme=0 for i in range(1,51): somme=somme+sqrt(i) print(somme)

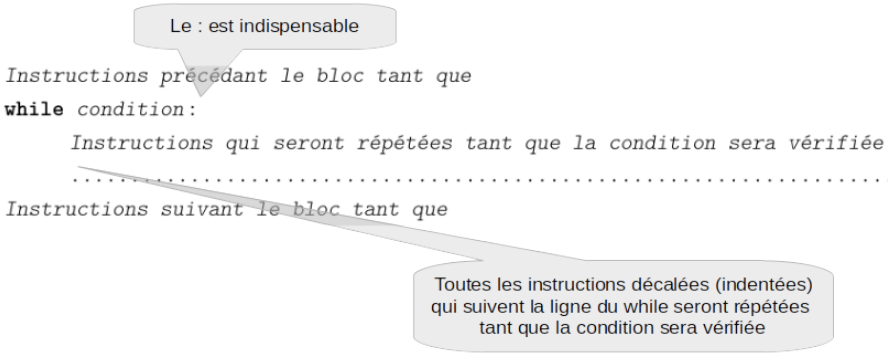
Boucles TANT QUE

• Rappel : ces boucles permettent de répéter des instructions tant qu'une certaine condition est vérifiée. La condition en question s'exprime en python de la même façon qu'avec un if.

Langage naturel

```
Instructions précédant le bloc tant que
TANT_QUE condition FAIRE
    Instructions qui seront répétées tant que la condition sera vérifiée
    ..........
    FIN_TANT_QUE
Instructions suivant le bloc tant que
```

Équivalent en python



► **Exemple :**

Script permettant de déterminer le premier entier dont le cube dépasse 5000.

Langage naturel	Équivalent en python

n ← 0
TANT_QUE $n^3 < 5000$ FAIRE
  n ← n+1
  FIN_TANT_QUE
AFFICHER n
	n=0
	while n**3<5000:
	  n=n+1
	print(n)

► Activité n°24

Compléter le code du script python ci-dessous pour qu'il corresponde à l'algorithme en langage naturel dont le but est de déterminer le premier entier n tel que le produit $1 \times 2 \times 3 \times \cdots \times n$ dépasse 10000 .

Langage naturel	Équivalent en python

n ← 1
produit ← 1
TANT_QUE produit<10000 FAIRE
  n ← n+1
  produit ← produit × n
  FIN_TANT_QUE
AFFICHER n
	n=1
	produit=1
	while .....
	  n=.....
	  produit=.....
	print(n)

► Activité n°25

Parmi les quatre scripts python ci-dessous, déterminer le seul valide indiquant le plus petit entier n tel que $0{,}9^n \leqslant 0.001$

Script 1	Script 2
n=0 while 0.9**n<=0.001: n=n+1 print(n)	n=0 while 0.9**n>0.001: n=n+1 print(n)
Script 3	**Script 4**

n=0
while 0.9**n<=0.001
  n=n+1
print(n)
	n=0
	while 0.9**n>0.001:
	  n=n+1
	print(n)

Les fonctions en programmation

• Rappel : une fonction en programmation est un bloc d'instructions qui ne sera exécuté que quand on l'appelle. Une fonction dépend en général d'un ou plusieurs paramètres et retourne une valeur dépendant de ces paramètres. L'utilisation d'une fonction n'a en général d'intérêt que si on l'appelle plusieurs fois.

Langage naturel

FONCTION mafonction(*paramètres séparés par des virgules*)
DEBUT_FONCTION
  Instructions
  RETOURNER...
FIN_FONCTION

Équivalent en python

Le : est indispensable
def mafonction(*paramètres séparés par des virgules*):
  Instructions
  return...
Toutes les instructions définissant la fonction doivent être décalées (indentées)

► Exemple :

Un cycliste et un piéton se déplacent à une vitesse respective de 18 $\text{km} \cdot \text{h}^{-1}$ et 6 $\text{km} \cdot \text{h}^{-1}$. Ce script donne le nombre de kilomètres parcourus par le cycliste et le piéton durant un même temps t (en heures).

Langage naturel	Équivalent en python

FONCTION distance(vitesse,temps)
DEBUT_FONCTION
  RETOURNER vitesse × temps
FIN_FONCTION
 	
DEBUT_ALGORITHME
LIRE t
AFFICHER distance(42,t)
AFFICHER distance(6,t)
FIN_ALGORITHME
	def distance(vitesse,temps):
	  return vitesse*temps
	
	t=float(input("t?"))
	print(distance(42,t))
	print(distance(6,t))

► Activité n°26

La formule de conversion entre la valeur d'une température en degrés farenheit et sa valeur en degrés celsius est la suivante :

$$\text{degrés Celsius} = \frac{5 \times (\text{degrés Fahrenheit}) - 160}{9}$$

Compléter le code du script python ci-dessous pour qu'il donne la valeur en degrés celsius d'une température de 100 et de 200 degrés farenheit.

Script python
def celsius(farenheit):   return ..... print(celsius(100)) print(.....)

3) Application

► Activité n°27
Compléter le code du script python ci-dessous pour qu'il corresponde à l'algorithme en langage naturel.

Langage naturel	Équivalent en python

POUR i ALLANT_DE 1 A 100
  AFFICHER i^3
  FIN_POUR
	for i .....
	  print(.....)

► Activité n°28
Que fait ce script python ?

Script python
```
from math import*
i=0
while i*sqrt(i)<500:
    i=i+1
print(i)
```

Réponse : .....
.....

► Activité n°29
La valeur d'une voiture était de 20 000 euros en 2018. On suppose qu'elle est multipliée par 0,9 chaque année. Compléter le script python ci-dessous pour qu'il indique la première année à partir de laquelle la valeur de la voiture sera inférieure ou égale à 12 000 euros.

Script python
```
annee=2018
valeur=20000
while ......................
    valeur=valeur*0.9
    annee=annee+1
print(annee)
```

► Activité n°30
Un ticket de tramway coûte 1 euro sans abonnement. Avec un abonnement annuel de 30 euros, le ticket ne coûte plus que 0,75 euros. Déterminer parmi les quatre scripts python ci-dessous le seul valide qui affiche, après avoir entré le nombre de tickets achetés `n`, si prendre un abonnement est rentable :

Script 1
```
n=int(input("n?"))
if 30+0.75*n<n:
    print("abonnement rentable")
else:
    print("abonnement pas rentable")
```

Script 2
```
n=int(input("n?"))
if 30+0.75*n<n:
    print("abonnement rentable")
else
    print("abonnement pas rentable")
```

Script 3
```
n=int(input("n?"))
if 30+0.75*n<n:
    print("abonnement rentable")
    else:
    print("abonnement pas rentable")
```

Script 4
```
n=int(input("n?"))
if 30+0.75*n>n:
    print("abonnement rentable")
else:
    print("abonnement pas rentable")
```

► Activité n°31
Durant une année un individu a acheté n DVD valant 15 euros pièce et p livres valant 22 euros pièce. Compléter le script python ci-dessous pour qu'il indique si cet individu a dépensé plus de 300 euros en dvd et livres ou non :

Script python
```
def depense_totale(n,p):
    return ........................

n=int(input("nombre de dvd?"))
p=int(input("nombre de livres?"))
if depense_totale(n,p).....................
    print("plus de 300 euros")
else:
    print("pas plus de 300 euros")
```

► Activité n°32
Un automobiliste roule à 90 $km \cdot h^{-1}$ pendant 2 heures, puis roule à 120 $km \cdot h^{-1}$. Compléter le script python ci-dessous pour qu'il indique le nombre de kilomètres parcourus en fonction du temps de parcours exprimé en heures :

Script python
```
def distance(temps):
    if temps<=2:
        return ........................
    else:
        return ........................

t=float(input("temps?"))
print (distance(t))
```

4) Autres fonctionnalités utiles de python

Chaines de caractères

On peut affecter à une variable une chaine de caractères en l'encadrant simplement par des guillemets. Exemple : `a="blabla"`

Situation (a et b : chaines)	Syntaxe python
mettre b au bout de a	a+b
connaitre la longueur de a	len(a)
récupérer le $(i+1)^{\text{ième}}$ caractère de a	a[i]

Exemple :

Code python	Résultat
a="bon" b="jour" print(a+b) print(len(a)) print(a[0]) print(b[3])	 bonjour 3 b r

Les listes

On peut affecter à une variable une liste de nombres (ou de chaines de caractères) en les encadrant par des crochets et en les séparant par des virgules. Exemple : `a=[1,3,5,7]`

► **Remarque :** en python, on commence à compter les éléments d'une liste à partir de la position 0. Par exemple, si `a=[1,3,5,7]` :

- le premier élément de la liste est `a[0]` (position 0) qui vaut donc 1 ;
- le deuxième élément de la liste est `a[1]` (position 1) qui vaut donc 3 ;
- etc.
- le dernier terme de la liste est `a[3]` (position 3) qui vaut donc 7 ;

Donc le terme à la position `i` d'une liste correspond à son $(i+1)^{\text{ième}}$ terme

Situation (a : liste)	Syntaxe python
connaitre le nombre d'éléments de a	len(a)
récupérer le terme de a à la position i	a[i]
ajouter l'élément e à la liste a	a.append(e)
supprimer le terme de a à la position i	del a[i]
insère l'élément e à la position i	a.insert(i,e)

► **Exemple :**

Code python	Résultat
a=[1,3,5,7] print(len(a)) print(a[3]) a.append(9) print(a) del a[1] print(a) a.insert(1,12) print(a)	4 7 [1, 3, 5, 7, 9] [1, 5, 7, 9] [1, 12, 5, 7, 9]

Tester des scripts python en ligne

Il est possible d'exécuter des scripts python depuis un navigateur sans aucune installation, ni inscription, ni connexion préalable à l'adresse suivante :
https://www.xm1math.net/SNT/python_en_ligne.html

3. Entraînement

► **Activité n°33**

Indiquer ce que va afficher chacun des scripts python suivants :

- Script 1 : ..... ; Script 2 : .....
- Script 3 : ..... ; Script 4 : .....

Script 1	Script 2

nombre=0
for i in range(1,11):
  nombre=nombre+1
print(nombre)
	nombre=0
	for i in range(1,11):
	  if i>5:
	  nombre=nombre+1
	print(nombre)

Script 3	**Script 4**

nombre=0
for i in range(1,11):
  if nombre>5:
  nombre=nombre+1
print(nombre)
	nombre=0
	for i in range(1,11):
	  if nombre<5:
	  nombre=nombre+1
	print(nombre)

► Activité n°34

Indiquer ce que va afficher chacun des scripts python suivants :

- Script 1 : ..... ; Script 2 : .....
- Script 3 : ..... ; Script 4 : .....

Script 1

```
for i in range(1,5):
    print(2*i)
```

Script 2

```
for i in range(1,6):
    print(i*i)
```

Script 3

```
n=1
for i in range(1,4):
    n=n+2
print(n)
```

Script 4

```
n=1
for i in range(1,4):
    n=n+2
    print(n)
```

► Activité n°35

Indiquer ce que va afficher chacun des scripts python suivants :

- Script 1 : ..... ; Script 2 : .....
- Script 3 : ..... ; Script 4 : .....

Script 1

```
n=43
while n>=10:
    n=n-10
print(n)
```

Script 2

```
n=43
while n>=10:
    n=n-10
    print(n)
```

Script 3

```
n=1
while n<20:
    n=n*4      print(n)
```

Script 4

```
n=1
while n<20:
    print(n)
    n=n*4
```

► Activité n°36

Compléter le script python suivant pour qu'il indique si les entiers a, b et c entrés sont rangés dans l'ordre croissant ou non.

Script python

```
a=int(input("a=?"))
b=int(input("b=?"))
c=int(input("c=?"))
if ............................ :
   print("dans l'ordre croissant")
else :
    print("pas dans l'ordre croissant")
```

► Activité n°37

Compléter le script python suivant pour qu'il indique lequel des entiers a, b et c entrés (et supposés différents) est le plus grand.

Script python

```
a=int(input("a=?"))
b=int(input("b=?"))
c=int(input("c=?"))
if a>b and a>c:
    print("a est le plus grand")
else:
    if ............................ :
       print("b est le plus grand")
    else:
       if ............................ :
          print("c est le plus grand")
```

► Activité n°38

On considère le script python ci-dessous :

Script python

```
a=int(input("a=?"))
b=int(input("b=?"))
tmp=b
b=a
a=tmp
print("a=",a)
print("b=",b)
```

1. Compléter le tableau suivant si on entre 1 comme valeur pour a et 5 comme valeur pour b :

Valeurs de a, b et tmp	a=1	b=5	tmp
après exécution de la 3e ligne			
après exécution de la 4e ligne			
après exécution de la 5e ligne			

2. Compléter le tableau suivant si on entre 5 comme valeur pour a et 1 comme valeur pour b :

Valeurs de a, b et tmp	a=5	b=1	tmp
après exécution de la 3e ligne			
après exécution de la 4e ligne			
après exécution de la 5e ligne			

3. Que fait ce script ? .....

► Activité n°39

On considère le script python ci-dessous dans lequel on a défini une fonction qui échange la valeur de 2 variables selon le principe vu à l'exercice précédent :

Script python

```
def echange(nombre1,nombre2):
    tmp=nombre2
    nombre2=nombre1
    nombre1=tmp
    return (nombre1,nombre2)

a=int(input("a=?"))
b=int(input("b=?"))
c=int(input("c=?"))
if a>b:
    (a,b)=echange(a,b)
if a>c:
    (a,c)=echange(a,c)
if b>c:
    (b,c)=echange(b,c)
print("a=",a)
print("b=",b)
print("c=",c)
```

1. Compléter le tableau suivant si on entre 3 comme valeur pour a, 2 comme valeur pour b et 1 comme valeur pour c :

Valeurs de a, b et c	a=3	b=2	c=1
après le bloc `if a>b:`			
après le bloc `if a>c:`			
après le bloc `if b>c:`			

2. Compléter le tableau suivant si on entre 2 comme valeur pour a, 3 comme valeur pour b et 1 comme valeur pour c :

Valeurs de a, b et c	a=2	b=3	c=1
après le bloc `if a>b:`			
après le bloc `if a>c:`			
après le bloc `if b>c:`			

3. Compléter le tableau suivant si on entre 1 comme valeur pour a, 3 comme valeur pour b et 2 comme valeur pour c :

Valeurs de a, b et c	a=1	b=3	c=2
après le bloc `if a>b:`			
après le bloc `if a>c:`			
après le bloc `if b>c:`			

4. Que fait ce script ? .....

► Activité n°40

On considère les scripts ci-dessous :

Script 1

```
somme=0
for i in range(1,21):
    somme=somme+i
print(somme)
```

Script 2

```
produit=1
for i in range(.....):
    produit=.....
print(produit)
```

1. Que va afficher le script 1 ?

A : 20
B : 21
C : le résultat de la somme $1+2+3+\cdots+19+20$
D : le résultat de la somme $1+2+3+\cdots+19+20+21$

2. Compléter le script 2 pour qu'il affiche le résultat du produit $1\times2\times3\times\cdots\times19\times20$.

► Activité n°41

Le PIB d'un pays émergent était de 700 milliards d'euros en 2020. Selon un modèle de prévision, il devrait augmenter de 10% par an dans les années futures.
Compléter le script python ci-dessous pour qu'il affiche en quelle année le PIB devrait avoir doublé de valeur par rapport à 2020.

Script python

```
annee=2020
PIB=700
while ..... :
    PIB=.....
    annee=annee+1
print(annee)
```

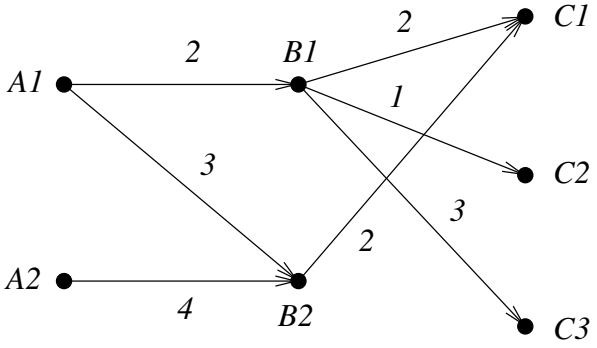
2

TP : Exemples de représentation matricielle

► **Activité n°42**

- Un petit site web A contient 2 pages $A1$ et $A2$;
- Un autre petit site web B contient 2 pages $B1$ et $B2$;
- Un dernier site web C contient 3 pages $C1$, $C2$ et $C3$.

Sur le schéma ci-dessous figure le nombre de liens qui pointent des pages du site A vers celles du site B et le nombre de liens qui pointent des pages du site B vers celles du site C.



1. Indiquer dans les matrices M et N ci-dessous, le nombre de liens qui pointent des pages du site A vers celles du site B et le nombre de liens qui pointent des pages du site B vers celles du site C.

$$M = \begin{matrix} \\ A1 \\ A2 \end{matrix}\begin{matrix} \begin{matrix} B1 & B2 \end{matrix} \\ \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} \end{matrix} \quad \text{et } N = \begin{matrix} \\ B1 \\ B2 \end{matrix}\begin{matrix} \begin{matrix} C1 & C2 & C3 \end{matrix} \\ \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \end{matrix}$$

2. Calculer la matrice MN. Que permet-elle de retrouver comme information ?

.....

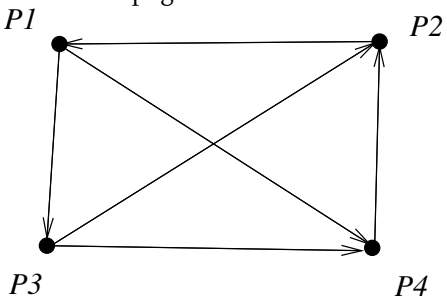
.....

.....

.....

► **Activité n°43**

Les flèches dans le schéma ci-dessous indique l'existence d'un lien direct (en un clic) entre les 4 pages d'un site web.



1. Peut-on partir de la page $P1$ et y revenir en cliquant sur deux liens ?
2. De combien de façons peut-on aller de la page $P1$ à la page $P2$ en cliquant sur deux liens ?
3. De combien de façons peut-on aller de la page $P1$ à la page $P3$ en cliquant sur deux liens ?
4. De combien de façons peut-on aller de la page $P1$ à la page $P4$ en cliquant sur deux liens ?
5. Construire M, la matrice carrée de dimension 4 de telle façon que le coefficient de la i^e ligne et de la j^e colonne soit égal à 1 s'il y a une flèche de P_i vers P_j et 0

dans le cas contraire.

$$M = \begin{array}{c} \\ \lceil \\ | \\ | \\ \lfloor \end{array} \begin{array}{cccc} P1 & P2 & P3 & P4 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{array} \begin{array}{c} \\ \rceil \\ | \\ | \\ \rfloor \end{array} \begin{array}{c} \\ P1 \\ P2 \\ P3 \\ P4 \end{array}$$

6. Calculer M^2. Que retrouve-t-on à la première ligne de M^2 ?

.....

.....

.....

.....

.....

► Activité n°44

Un journal propose deux formules d'abonnement : une formule « papier » classique notée A et une formule 100% numérique notée B. En 2020 la répartition des abonnés est de 50% pour la formule A et de 50% pour la formule B.

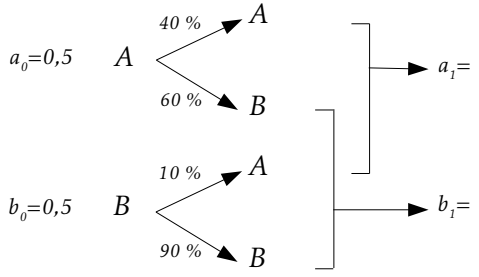
Le journal prévoit que d'une année sur l'autre (à effectif d'abonnés stable) :

- 60% des abonnés à la formule A passeront à la formule B ;
- 10% des abonnés à la formule B passeront à la formule A .

On note :

- a_0 la proportion d'abonnés à la formule A en 2020 ;
- b_0 la proportion d'abonnés à la formule B en 2020 ;
- a_1 la proportion d'abonnés à la formule A en 2021 ;
- b_1 la proportion d'abonnés à la formule B en 2021 ;
- a_2 la proportion d'abonnés à la formule A en 2022 ;
- b_2 la proportion d'abonnés à la formule B en 2022 ;
- ainsi de suite...

a) Indiquer ci-dessous, le calcul de a_1 et b_1.



b) En déduire les coefficients à inclure dans la matrice ci-dessous pour que le calcul soit correct :

$$(a_1\ b_1) = (a_0\ b_0)\begin{pmatrix} \cdots & \cdots \\ \cdots & \cdots \end{pmatrix}.$$

c) Le mécanisme étant le même d'une année sur l'autre, pour avoir la répartition entre les formules d'abonnement en 2022, il suffit de remultiplier par la même matrice. Compléter le calcul ci-dessous :

$$(a_2\ b_2) = (a_1\ b_1)\begin{pmatrix} \cdots & \cdots \\ \cdots & \cdots \end{pmatrix} = (\ldots\ldots\ \ldots\ldots)\begin{pmatrix} \cdots & \cdots \\ \cdots & \cdots \end{pmatrix} = (\ldots\ldots\ \ldots\ldots)$$

d) On cherche à programmer un script python qui donne l'évolution de la répartition jusqu'en 2030. Expliquer pourquoi seul le script 2 ci-dessous pourra répondre au problème.

Script 1

```
a=0.5
b=0.5
for n in range(1,11):
    a=0.4*a+0.1*b
    b=0.6*a+0.9*b
    print("Année ",2020+n," : a=",a," b=",b)
```

Script 2

```
a=0.5
b=0.5
for n in range(1,11):
    a=0.4*a+0.1*b
    b=1-a
    print("Année ",2020+n," : a=",a," b=",b)
```

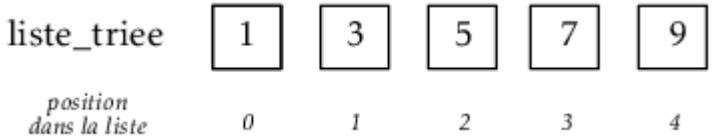
3 TP : Traitement des données

1. Trier des données numériques

Pour pouvoir exploiter des données numériques, une des opérations standards à maitriser est le tri de ces données. Nous allons voir le principe d'une des méthodes les plus classiques pour cela, qui s'appelle le *tri par insertion*.

1) 1re étape : comment insérer un nouvel élément dans une liste triée ?

► **Exemple :** on considère la liste de nombres déjà triés dans l'ordre croissant ci-dessous que l'on appelle `liste_triee`.



Dessous la liste on a fait figurer la position des éléments dans la liste en commençant par 0 comme souvent en informatique.

a) Si l'on veut insérer comme nouvel élément le nombre 4 dans la liste en respectant l'ordre croissant, à quelle position faudrait-il l'insérer ? .....

b) Si l'on veut insérer comme nouvel élément le nombre 8 dans la liste en respectant l'ordre croissant, à quelle position faudrait-il l'insérer ? .....

c) On cherche maintenant un moyen algorithmique (donc automatique) pour savoir à quelle position placer n'importe quel nouveau élément. Pour cela on se place au début de la liste et on va chercher un principe qui répond au problème sous la forme suivante :
`tant qu'on est pas à la fin de la liste et que le terme de la liste à la position courante est ..... au nouvel élément que l'on veut insérer, on passe à la position suivante`
Compléter la phrase pour que la méthode soit valable.

d) On va maintenant pouvoir en déduire une fonction python qui prend en paramètres une liste de nombres déjà triée dans l'ordre croissant et un nouvel élément et qui va retourner une nouvelle liste toujours triée mais à laquelle on a inséré à la bonne place le nouvel élément :

```
def insere(liste_triee,nouvel_element):
    position=0
    while (position<len(liste_triee) and liste_triee[position]..nouvel_element):
        position=position+1
    liste_triee.insert(position,nouvel_element)
    return liste_triee
```

Compléter l'inégalité dans la condition de la boucle `while` pour que le code corresponde à la méthode vue à la question précédente.

2) Le principe du tri par insertion

► Explication du tri par insertion avec un exemple :
Pour trier cette liste :

- On initialise une nouvelle liste (`liste_triee`) avec comme premier élément celui de la liste à trier :

- Puis on insère au fur et à mesure dans `liste_triee`, avec la méthode vue au paragraphe précédent, tous les autres termes de la liste à trier. Ainsi, on va insérer, en respectant l'ordre croissant, à `liste_triee` le 3, puis le 7, etc.

Cela donne en python (en affichant toutes les étapes) :

Code python

```
def insere(liste_triee,nouvel_element):
    position=0
    while (position<len(liste_triee) and
liste_triee[position]<nouvel_element):
        position=position+1
    liste_triee.insert(position,nouvel_element)
    return liste_triee

liste_non_triee=[12,3,7,1,9,24,8]
liste_triee=[liste_non_triee[0]]

for position in range(1,len(liste_non_triee)):
    insere(liste_triee,liste_non_triee[position])
    print(liste_triee)
```

Résultat après exécution

```
[3, 12]
[3, 7, 12]
[1, 3, 7, 12]
[1, 3, 7, 9, 12]
[1, 3, 7, 9, 12, 24]
[1, 3, 7, 8, 9, 12, 24]
```

► Activité n°45
Quelle modification au code faudrait-il apporter pour n'afficher que le résultat final ?
.....
.....
.....

► Remarque : Le principe du tri par insertion est celui qu'on utilise naturellement quand on range des cartes au fur et à mesure qu'elles sont distribuées.

2. Organisation des données

On vient de voir comment trier une donnée simple (une liste d'entiers), mais beaucoup de données en informatiques sont en fait composées de plusieurs éléments liés entre eux (nombres, textes, images, sons, etc.) qu'il faut organiser de façon à pouvoir les exploiter et éventuellement les croiser.

Info sur les données structurées

Pour illustrer le vocabulaire, on va utiliser les données météo suivantes :

Heure	Température °C	Vent km/h
00h	22.4	4
02h	19.9	11
04h	19.2	9

- Les données fournies dans ce tableau constituent ce qu'on appelle en informatique une ..... .
- Elle contient trois types de données (dites données .....)

  classées par colonnes.
  On dit en informatique que l'heure, la température et la vitesse du vent sont des ..... ou ..... de la table.

- Il existe plusieurs formats de fichiers pour enregistrer ce type de table de données. L'un des plus simples (et sommaires) est le format *csv* (« ..... ») qui consiste en un

  simple fichier texte dont la première ligne contient les intitulés des champs de la table séparés par des virgules et où ensuite chaque ligne contient les valeurs séparées elles aussi par une virgule.

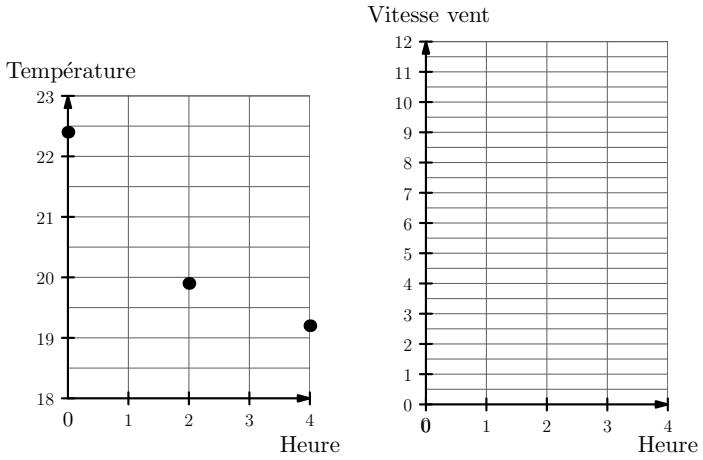
- Ainsi, le fichier *csv* correspondant à l'exemple serait constitué de la façon suivante :

```
Heure,Température °C,Vent km/h
00h,22.4,4
02h,19.9,11
...,...,...
```

  Compléter la dernière ligne du fichier.
- Les algorithmes de tri permettent alors de trier la table de données selon le *champ* que l'on veut.
  Ci-dessous, figure la même table que l'exemple mais trié selon un autre champ. Lequel ? .....

Heure	Température °C	Vent km/h
00h	22.4	4
04h	19.2	9
02h	19.9	11

- On peut aussi extraire les données d'une table pour créer des graphiques. Ci-dessous, figurent le graphique complet pour les températures et le repère pour celui de la vitesse du vent :



  Rajouter les points pour la vitesse du vent.

3. Cryptage des données - Initiation

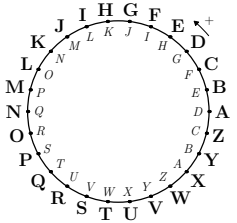
1) Vocabulaire de base

- L'action de transformer un texte compréhensible par tous en un autre incompréhensible (sauf pour le destinataire autorisé) s'appelle ..... ;
- L'action inverse permettant au destinataire autorisé de reconstituer le texte original s'appelle ..... ;
- L'action consistant par une personne non autorisée à retrouver le texte original s'appelle ..... ;

2) Le chiffre de César

Une des premières méthodes pour chiffrer a consisté à remplacer les caractères du texte par d'autres dans le même alphabet (.....).

Le principe général de la méthode utilisée par Jules César consiste à remplacer chaque lettre par la lettre située située à un certain rang fixe plus loin dans l'alphabet. Pour un décalage de 3 rangs (utilisé historiquement par Jules César), la situation peut se représenter selon le schéma circulaire suivant (à l'extérieur se trouve la lettre d'origine et à l'intérieur la lettre de substitution) :



► Activité n°46

`DWWDTXHDPLGL` est le texte chiffré par la méthode de César avec un décalage de 3 rangs. Quel était le texte original ?
Réponse : .....

► Activité n°47

1. Pour un alphabet ne comportant que les lettres en majuscule de `A` à `Z`, combien de décalage sont possibles avec la méthode de chiffrement de César ?
   Réponse : .....
2. `YLUMVYAZCPLUKYVUAKLTHPU` est un message chiffré par la méthode de César avec un décalage inconnu. Quel était le texte original ?
   Réponse : .....

► Codage en python possible pour le (dé)chiffrement de César :

Script python

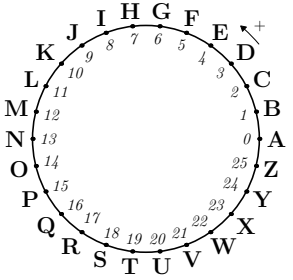
```
def chiffrer(message_en_clair,cle):
      message_chiffre=""
      for i in range(len(message_en_clair)):
            message_chiffre+=chr((ord(message_en_clair[i])-65+cle)%26+65)
      return message_chiffre

def dechiffrer(message_chiffre,cle):
      message_dechiffre=""
      for i in range(len(message_chiffre)):
            message_dechiffre+=chr((ord(message_chiffre[i])-65-cle)%26+65)
      return message_dechiffre
```

3) Le chiffre de Vigenère

La méthode de chiffrement de Vigenère est aussi un chiffrement par substitution mais avec un décalage non fixe.
Principe :

- Le chiffre de Vigenère utilise un mot clé que l'on répète (en dessous du texte à chiffrer) autant de fois que nécessaire afin d'avoir la même longueur que le message à chiffrer. Exemple avec comme message LEMESSAGEACRYPTER et comme clef MACLEF :
  LEMESSAGEACRYPTER
  MACLEFMACLEFMACLE
- La lettre du mot clé figurant dans la deuxième ligne indique le décalage à apporter à la lettre du message située au dessus. Exemple :
  — Si la lettre de la clef est un A, on ne décale pas la lettre correspondante du message ;
  — Si la lettre de la clef est un B, on décale d'un rang dans l'alphabet la lettre correspondante du message : ainsi un A dans le message devient un B, un B devient un C, etc...
  — Si la lettre de la clef est un C, on décale de 2 rangs dans l'alphabet la lettre correspondante du message : ainsi un A dans le message devient un C, un B devient un D, etc...
  — etc...
- Avec notre exemple : la première lettre du message est un L ; la lettre correspondante de la clef est un M qui représente un décalage de 12. La première lettre du message crypté est donc un X (12 lettres plus loin que L dans l'alphabet).

► **Activité n°48**
On reprend l'exemple précédent :
LEMESSAGEACRYPTER
MACLEFMACLEFMACLE
Les 10 premières lettres du message crypté sont XEOPWXMGGL.
Donner les sept dernières lettres (correspondantes à CRYPTER) :
Réponse : .....
On pourra s'aider de l'alphabet en cercle ci-dessous :



► **Activité n°49**
Pour le déchiffrement dans le procédé de Vigenère, on applique le même principe que pour le chiffrement sauf que le décalage pour le déchiffrement est égal à 26 moins le décalage pour le chiffrement, ce qui revient à tourner dans le sens contraire sur le cercle de l'exercice précédent.
Ci-dessous figure un message crypté avec toujours comme clef MACLEF :
DDXLQNPI
MACLEFMA
Déchiffrer le message. Réponse : .....

► Codage en python possible pour le (dé)chiffrement de Vigenère :

Script python

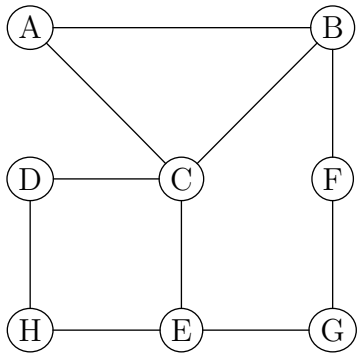
```
def chiffrer(message_en_clair,cle):
      message_chiffre=""
      for i in range(len(message_en_clair)):
            decalage=ord(cle[i%len(cle)])-65
            message_chiffre+=chr((ord(message_en_clair[i])-65+decalage)%26+65)
      return message_chiffre

def dechiffrer(message_chiffre,cle):
      message_dechiffre=""
      for i in range(len(message_chiffre)):
            decalage=26-(ord(cle[i%len(cle)])-65)
            message_dechiffre+=chr((ord(message_chiffre[i])-65+decalage)%26+65)
      return message_dechiffre
```

4 TP : Exemple de table de routage

Dans un réseau de routeurs, la table de routage indique, pour chaque routeur, comment envoyer des données à un autre routeur.

► Activité n°50

On considère le réseau de routeurs et sa table de routage ci-dessous :



Routeur A		Routeur B		Routeur C		Routeur D	
Pour	Passer par	Pour	Passer par	Pour	Passer par	Pour	Passer par
B	B	A	A	A	A	A	C
C	C	C	C	B	B	B	C
D	C	D	C	D	D	C	C
E	C	E	C	E	E	E	H
F	B	F	F	F	B	F	C
G	C	G	F	G	E	G	H
H	C	H	C	H	E	H	H

Routeur E		Routeur F		Routeur G		Routeur H	
Pour	Passer par	Pour	Passer par	Pour	Passer par	Pour	Passer par
A	C	A	B	A	F	A	D
B	C	B	B	B	F	B	D
C	H	C	G	C	E	C	D
D	H	D	B	D	E	D	D
F	G	E	G	E	E	E	E
G	G	G	G	F	F	F	E
H	H	H	G	H	E	G	E

1. Quel parcours doivent suivre des données pour aller du routeur E au routeur G ?
   Réponse : .....
2. Quel parcours doivent suivre des données pour aller du routeur E au routeur F ?
   Réponse : .....
3. Quel parcours doivent suivre des données pour aller du routeur A au routeur G ?
   Réponse : .....
4. Quel parcours doivent suivre des données pour aller du routeur F au routeur C ?
   Réponse : .....