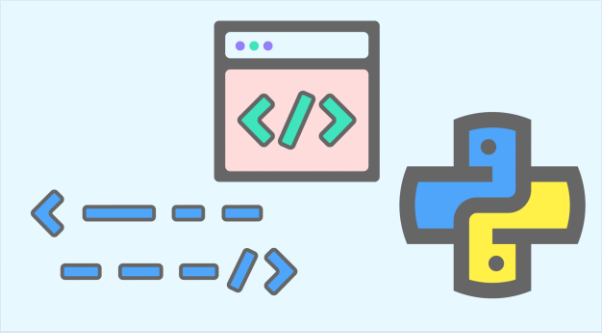


Manuel de l'élève

ALGORITHME

Matière: Informatique

3ème Math

KAOUTHER ALLEGUI

Sommaire

Partie 1 : Démarche de résolution de problèmes

I.Introduction :

L'ordinateur est une machine électronique utilisé presque dans tous les domaines de vie pour réaliser des différents types de traitements grâce à de programmes enregistrés dans sa mémoire. Ces programmes sont élaborés par des informaticiens et pour les réaliser il y a toute une démarche à suivre commençant par l'énoncé du problème jusqu'à abouti à une solution exécutable sur machine.

II.Les étapes de résolution d'un problème

Activité 1 :

On se propose de calculer et d'afficher la surface **S** et le périmètre **P** d'un rectangle de longueur **Lo** et de largeur **La**.

1. Ecrire un algorithme permettant de résoudre ce problème.
2. Implémenter cet algorithme en Python.

1. L'algorithme

Un **algorithme** est une suite finie ordonnée (et logique) d'opérations (d'instructions / actions) écrites dans un ordre chronologique bien déterminé afin de résoudre un problème donné.

Structure générale d'un algorithme

Algorithme nom_algorithme
Début
…….
……. } Les différentes instructions qui obéissent aux conventions et aux notations algorithmiques

fin nom_algorithme

Déclaration des objets

Objet	Type / Nature

Remarques :

- Le nom_algorithme est généralement un nom qui donne une idée sur ce que fait l'algorithme. Par exemple, si on écrit un algorithme qui calcule la somme de 2 entiers, on attribue à notre algorithme le nom **somme**.
- Chaque algorithme possède un nom et commence par le mot « **Début** » comme il finit par le mot «**Fin**»
- Le verbe « **Lire** » est utilisé pour la saisie d'une donnée.
- Le verbe « **Ecrire** »est une notation utilisée pour afficher un objet sur l'écran.
- L'utilisation du signe « ← **(reçoit)** » pour l'affectation d'une valeur ou une expression à une variable.

Exp : x ← 5 on dit : x reçoit 5

2. Programme :

L'algorithme est la solution d'un problème informatique dans un langage naturel. Cette solution n'est pas compréhensible par l'ordinateur. Pour cela elle doit être traduite en **un langage de programmation**, ce qui donne **un programme.**

Un **langage de programmation** est un ensemble de **règles syntaxiques** (grammaire et vocabulaire) et **sémantiques** (étude du sens des mots) qui sert à **transformer** un **algorithme** en un **programme source** pour que l'ordinateur puisse l'exécuter.

On distingue deux types de langages évolués : les langages interprétés et les langages compilés.
Exemples :

Interprétés	Compilés
APL, Python, Basic	Algol, C , C++, Pascal

Pour le langage interprété, le programme source est exécuté **instruction par instruction.**
Dans le cas d'un langage compilé, le programme source est compilé (les erreurs sont corrigées) pour produire un autre programme dit exécutable.

Le langage de programmation qu'on va utiliser est le : **PYTHON**

Vous pouvez télécharger IDLE Python à partir de ce lien :https://www.python.org/downloads/
Python est un langage de programmation interprété multi-paradigme et multiplateformes.

Solution :

<table>
<tr><th>Algorithme</th><th>Implémentation en Python</th></tr>
<tr><td>
Algorithme ……
Début
……
……
……
……
……
……
……
Fin

T.D.O : (Tableau de déclaration des objets)

<table>
<tr><th>Objet</th><th>Type/Nature</th></tr>
<tr><td>…….</td><td>……..</td></tr>
<tr><td>……</td><td>……..</td></tr>
<tr><td>……</td><td>……..</td></tr>
<tr><td>……</td><td>……..</td></tr>
</table>
</td><td>
……
……
……
……
……

Résultat d'exécution
<pre>Taper Lo :2.5
Taper La :1.25
La surface = 3.125 Le périmètre = 7.5</pre>
</td></tr>
</table>

Partie 2 : Les opérations élémentaires simples

I.L'opération d'entrée :

1. Définition :

L'opération d'entrée c'est l'instruction qui permet à l'utilisateur de rentrer ou de saisir des valeurs au clavier pour qu'elles soient utilisées par le programme

2. Vocabulaire et syntaxe :

En algorithme	En Python
Lire (nom_variable)	nom_variable =input('donner la valeur')

3. Exemples

En algorithme	En Python
Lire (nom) Lire(n) Lire(x)	nom = input ( ) n = int (input( )) x = float (input( ))

4. Remarque :

♣ Quand on demande à la machine de lire une variable, cela implique que l'utilisateur va devoir écrire cette valeur.

♣ Les données qui sont lues doit être compatibles aux variables réservées en mémoire.

II. L'opération de sortie :

1. Définition :

L'opération de sortie c'est l'instruction qui permet au programme de communiquer des valeurs à l'utilisateur en les affichant à l'écran.

2. Exemple :

En Algorithme	En Python
Affichage d'un texte : Ecrire ("La valeur de n est : ")	print ("La valeur de n est : ")
Affichage de contenu d'une variable : Ecrire (n)	print (n)
Affichage mixte (texte et variable) : Ecrire ("La valeur de n est : ", n)	print ("La valeur de n est : ", n)
Affichage avec retour à la ligne : Ecrire_nl ("La valeur de n est : ", n)	print ("La valeur de n est : ", n, "\n")

3. Remarque :

♣ Quand on demande à la machine d'écrire une valeur, c'est pour que l'utilisateur puisse la lire.

4. Application :

Ordonner ces instructions pour que l'algorithme affiche le montant m à payer par un client qui a acheté n cahiers sachant que le prix du cahier est 2500 millièmes et qu'il a une remise r de 10%.

N° d'instruction	Instructions
…….	Ecrire ("Le montant payé est: ", m)
…….	m← 2500 * n
…….	Ecrire ("Donner la quantité : "), Lire (n)
……	r ←(10*m)/100
…….	m ← m-r

En déduire le Tableau de Déclaration des Objets (TDO)

Objet	Type/Nature
……	……
…….	…….
…….	……

III. L'opération d'affectation :

1. Définition :

L'opération d'affectation c'est une action qui permet d'affecter une valeur à une variable. Elle est représentée par une flèche orientée vers la gauche « ← ».

2. Vocabulaire et syntaxe :

En Algorithme	En Python
Variable← valeur	Variable = valeur

3. Exemple:

En Algorithme	En Python	Résultat
x←5	**x=5**	x contient 5
a← x+3	**a= x+3**	a contient 8
x←x-2	**x=x-2**	x contient 3

4. Remarque :

♣ Le type des variables situées à droite doit être de même type ou de type compatible que celle située à gauche.

Application 1 :

Soit la séquence d'affectations suivante :

1) x← 10
2) y ← 8
3) z ← x
4) x ← y
5) y ← z

1. Donner le résultat d'exécution de cette séquence sous forme d'un tableau.

N° de l'instruction	1	2	3	4	5
x					
y					
z					

2. Quelles sont les valeurs finales de x et de y ?

.....

3. Quel est le rôle de cette séquence ?

.....

4. Quelle est l'utilité de la variable z ?

.....

Application 2 :

1. Compléter le tableau suivant :

Instruction	Valeur de A	Valeur de B
A←5		
B←7		
A←A+B		
B←A-B		
A←A-B		

2. Quel est le rôle cet ensemble d'instructions ?

.....

Application 3 :

Ecrire un algorithme et son implémentation en Python d'un programme qui permet de permuter les contenus de deux réels a et b.

Solution :

<table>
<tr><th>Algorithme</th><th>Implémentation en Python</th></tr>
<tr><td>
Algorithme……..
Début
……
……
……
……
……
……
……
Fin
T.D.O :
<table>
<tr><th>Objet</th><th>Type/Nature</th></tr>
<tr><td>…….
…….
…….</td><td>……
……
……</td></tr>
</table>
</td><td>……
……
……
……
……
……
……</td></tr>
</table>

Partie 3 : Les structures de données

I. Les objets :

Dans un programme informatique, on va avoir en permanence besoin de stocker provisoirement des valeurs. Il peut s'agir de données issues du disque dur, fournies par l'utilisateur (frappées au clavier). Ces données peuvent être de plusieurs types : elles peuvent être des nombres, du texte,…

Activité1 :

On veut écrire un programme qui permet de calculer et d'afficher la surface d'un cercle de rayon R

```
from math import *
r=int(input('donner un entier a  '))
s=pi*r*r
print('la surface est egale',s)
```

```
donner un entier a  2
la surface est egale 12.566370614359172
```

Constatations :

Un programme manipule « des objets » qui peuvent changer de valeurs (appelés « variables » et d'autres objets qui ne changent pas de valeurs (appelés des « constantes »)

II. Les constantes et les variables :

1. Les constantes :

Définition :

Une constante est un objet dont la valeur est fixe au cours de l'exécution du programme.
Une constante est caractérisée par :

- **Son nom**
- **Sa valeur**

Déclaration :

Tableau de Déclaration des Objets :

Objet	Type/Nature
Nom_const	Constante de valeur

Exemple : pi=3.14

Tableau de Déclaration des Objets :

Objet	Type/Nature
pi	Constante de valeur 3,14

Activité 2:

Ecrire un code en Python qui permet :

- D'afficher la constante pi,
- D'incrémenter pi de 1
- D'afficher pi
- Exécuter ce code, que constatez-vous?

```
from math import *
print(pi)
pi=pi+1
print(pi)
```

```
3.141592653589793
4.141592653589793
>>
```

Constatations:

On constate que la constante ……peut être changée au cours du programme. Mais sa valeur initiale reste la ……

Après la fermeture de l'éditeur IDLE et son ouverture de nouveau, on remarque que la valeur pi est à ……

2. Les variables :

Définition :

Une variable est une case mémoire dans laquelle une valeur est **enregistrée** ou **stockée**. Cette valeur peut être modifiée tout au long du programme.

Une variable est caractérisée par :

- **Son nom**
- **Son type**
- **Son contenu**

Déclaration :

Tableau de Déclaration des Objets :

Objet	Type/Nature
Nom_var	Type_var

Exemple : A←2

On définit la variable« a » avec sa valeur 2

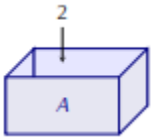


Tableau de Déclaration des Objets :

Objet	Type/Nature
A	entier

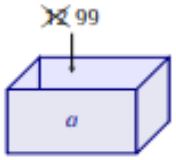
Remarque : le contenu d'une variable peut être modifié soit par une opération d'affectation soit par une opération de lecture (lire (a))

Une fois une variable initialisée, on peut modifier sa valeur en utilisant de nouveau l'opérateur d'affectation (=). La valeur actuelle de la variable est remplacée par la nouvelle valeur qu'on lui affecte. Dans l'exemple suivant, on initialise une variable à la valeur 12 et on remplace ensuite sa valeur par 99 :

a ← 12

a ← 99

ecire (a)



Activité 1 :

On se propose de calculer et d'afficher sur l'écran le périmètre P et la surface S d'un cercle de rayon R. Pour ce fait, on vous demande d'écrire l'algorithme correspondant et son implémentation en Python.

Solution

Algorithme	Implémentation en Python
Algorithme ……	
Début	…….
…….	…….
…….	…….
…….	…….
…….	…….
…….	
Fin	
T.D.O : (Tableau de déclaration des objets)	
Objet / Type/Nature …… / …… …… / …… …… / ……	

Remarque :

- Dans python les variables n'ont pas besoin d'être déclarées à l' avance, mais toute variable doit avoir une valeur avec le signe d'affectation =.
- Une variable n'a pas de type unique
- On obtient le type d'un objet o avec la fonction type(o) et son identité avec id(o) c'est un entier unique associe à tout objet : son adresse en mémoire
- id() : renvoie un entier représentant l'identifiant interne d'un objet
- type() : renvoie le type d'un objet.
- dir() : liste l'ensemble des fonctionnalités d'un objet.
- Commentaires : Tout texte qui suit le caractère dièse « # » n'est pas exécuté par Python mais sert à expliquer le programme.

Nomenclature

En Python, une variable :

- Le nom d'une variable s'écrit avec des lettres de a..z ou de A..Z (non accentuées), des chiffres ou bien tiret bas_
- Les accents sont possibles mais à éviter
- Les mots clé du langage sont interdits
- Le nom d'une variable ne doit pas commencer par un chiffre.
- ne contient pas des caractères spéciaux pas d'espaces on peut utiliser des underscores '_'
- doit commencer par une lettre ou un Underscore« _ »(Exp :age _age)
- contient des chiffres
- ne contient pas de caractères spéciaux (@ - ; ? ! …)
- ne contient pas d'espaces
- peut avoir des underscores pour les noms de variables combinés (Exp :age_personne)

Attention !

Python distingue les majuscules des minuscules (Python est sensible à la casse)

Donc age, AGE ou Age sont des variables différentes.

Exp :

x1 est un identificateur valide

_x1 est un identificateur ……..

1x_y n'est pas un identificateur valide car il commence par un …….

x-y……..

Exemples d'identificateurs

– **valides** : toto, proch_val, max1, MA_VALEUR, r2d2, bb8, _mavar

– **non valides** : 2be, C-3PO, ma var

Conventions : pour les variables en Python :

- utiliser des minuscules
- pas d'accents

III. Les types de données :

Il caractérise les valeurs que peut prendre cette donnée ainsi que les opérations autorisées sur celle-ci

1. Le type Entier (int) :

Définition :

Le type Entier désigne un sous ensemble des nombres entiers relatifs Z.

Déclaration :

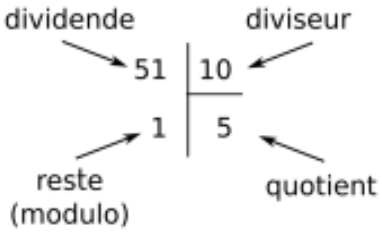
Objet	Type/Nature
Nom_variable	entier

Exemple :

Objet	Type/Nature
a	entier

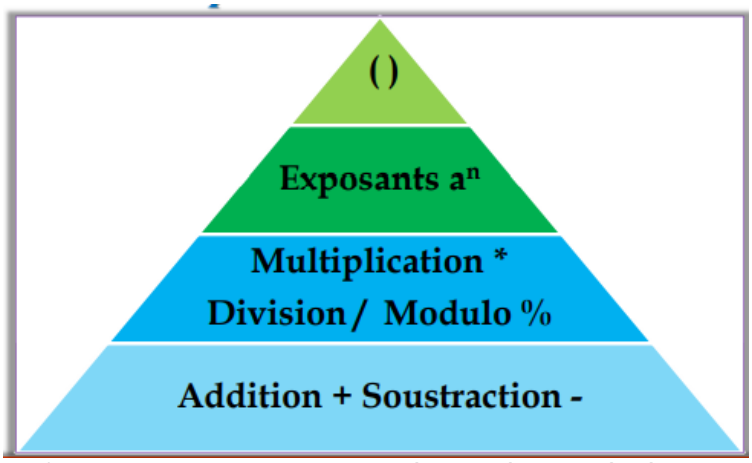
Les opérations arithmétiques et relationnelles sur les entiers :

Opérations (en Algorithme)	Opérations (en Python)	Rôles	Exemples
+	+	Adition	5 + 2 = ...
-	-	Soustraction	5 – 2 = ...
*	*	Multiplication	5 * 2 = ...
/	/	Division	5 / 2 = ...
=	= =	Egalité	5 = = 5 renvoie .....
<	<	Inférieur	2 < 5 renvoie .....
>	>	Supérieur	5 > 2 renvoie .....
≤	<=	Inférieur ou égal	2 <= 5 renvoie .....
≥	>=	Supérieur ou égal	5 >= 2 renvoie .....
≠	!=	Différent	5 != 2 renvoie .....
n ∈ [5,20]	5<=n<=20	Appartient	5<=3<=20 renvoie .....
X^n	X ** n	Puissance	5 ** 2 = ...
MOD	%	Modulo	5 % 2 = ...
DIV	//	Division entière	5 // 2 = ...



Ordre de priorité des opérateurs :

L'ordre de priorité est :
1. les parenthèses ()
2. la multiplication *, la division /, la division entière // et le reste de la division entière %
3. L'addition + et la soustraction –



N.B. Si les opérateurs ont même priorité, on commence de gauche vers la droite

Application :

On se propose de saisir un nombre t en seconde et de l'affiche en heure h, minute mn et seconde s. Pour ce fait, on vous demande de d'écrire l'algorithme correspondant et son implémentation en Python.
Exemple : t=4000s le programme affiche : 1 h 6 mn 40 s

Solution :

Algorithme	Implémentation en Python
Algorithme …….. **Début** …… …… …… …… …… .**Fin** **T.D.O : (Tableau de déclaration des objets)** **Objet** / **Type/Nature** ……. / …….. …… / …….. …….. / …….. …… / ……	…… …… …… …… …… ……

2. Le type Réel (float) :

Définition :

Le type Réel désigne un sous ensemble des nombres Réels IR.

Déclaration :

Objet	Type/Nature
Nom_variable	entier

Exemple :

Objet	Type/Nature
x	réel

Les opérateurs : les mêmes que ceux du type entier.

Exercice :

Compléter le tableau suivant :

Instruction algorithmique	En python	Valeur de x
x← 15 + 3 * 2 + 5	……	……
x← (18 mod 5) / 2	……	……
x ← (3 mod 5) div 2	……	……
x ← abs(-12.5)	……	……
x← non("bonjour" ≥ "Bonjour")	……	……
x ← 1=2	……	……
x← (10 ≠ (9+1)) ou (12 > -1)	……	……

Les fonctions prédéfinies sur les entier et les réels :

Nom en algorithme	Nom en Python	Rôle	Exemples
y←arrondi (x)	y=**round(x)**	Retourne l'entier le plus proche de la valeur de x	y←arrondi (9.499) Y contient ……. y←arrondi (2.5) Y contient…… y←arrondi (3.5) Y contient ……. y←arrondi (8.99) Y contient…….
y←abs (x)	y=**abs(x)**	Retournela valeur absolue de x	y←abs (-5) Y contient …….
y←racine carre (x)	from math import * print(sqrt(4))	Retournela racine carré de x (si x est positif)	y←racinecarre(4) Y contient …….
y←alea (a,b)	from random import * y=randint(a,b)	Retourne un entier entre [a et b]	Y←alea(2,5) Y contient ……
y←ent (x)	y=int(x)	Retourne la partie entière de x	y←ent (3) Y contient…….. y←ent (3.22) Y contient ……..

Application : Ecrire un algorithme d'un programme qui permet de calculer puis d'afficher la distance d entre deux points A(x1, y1) et B(x2, y2) sachant que d(A,B) = $\sqrt{(x2-x1)^2+(y2-y1)^2}$

Algorithme	Implémentation en Python
Algorithme …… **Début** …… …… …… …… …… …… …… …… **Fin**	…… …… …… …… …… …… …… ……

T.D.O : (Tableau de déclaration des objets)

Objet	Type/Nature
……	……

Application :

Ecrire un algorithme intitulé CHIFFRE permettant de saisir un nombre « N» formé de 3 chiffres, de calculer et d'afficher la somme S et le produit P des chiffres formant « N »

Exemple : N= 265

le programme affichera : La somme des chiffres est = 13(2+6+5 = 13)

Le produit des chiffres est = 60(2*6*5= 60)

Algorithme	Implémentation en Python
Algorithme …… **Début** …… …… …… …… …… …… …… **Fin**	…… …… …… …… …… …… ……

T.D.O : (Tableau de déclaration des objets)

Objet	Type/Nature
……	……

3. Le type Booléen (bool) :

Définition :

Le type booléen comporte deux valeurs Vrai et Faux (True et False en Python).

Déclaration :

Objet	Type/Nature
Nom_variable	booléen

Exemple :

Objet	Type/Nature
b	booléen

Opérations logiques sur les Booléens :

Opérateur		Signification
Algorithme	Python	
NON	not	Négation(le contraire)
ET	and ou &	Conjonction
OU	or ou |	Disjonction

La table de vérité qui manipule ces opérateurs est :

P	Q	Non(P)	P ET Q	P OU Q
Faux	Faux	Vrai	Faux	Faux
Faux	Vrai	Vrai	Faux	Vrai
Vrai	Vrai	Faux	Vrai	Vrai
Vrai	faux	Vrai	Faux	Vrai

Application : Compléter le tableau suivant :

Instruction	Résultat	Type
A ← 5+7		
B← 5-7		
C← 5*7		
D←5/7		
E←5 div 7		
F←5 mod 7		
G←5>7		
H←5≠7		
A ←(2>7) et (-2<5)		
B←(Alea [1,3]<10) ou (4>2)		
C←non (10≠-2)		

4. Le type Caractère (str):

Définition :

Un caractère (chiffre ou lettre ou symbole) est représenté le caractère lui-même mis entre guillemets
Les caractères sont les lettres (minuscules [" a " .. "z"] et majuscules [" A " .. " Z "]), les chiffres [" 0 " .." 9 "] et les caractères spéciaux : ponctuation, signes et mêmes les caractères non imprimables comme : Retour chariot, , Bip sonore,échappe, l'espace…

Remarque :

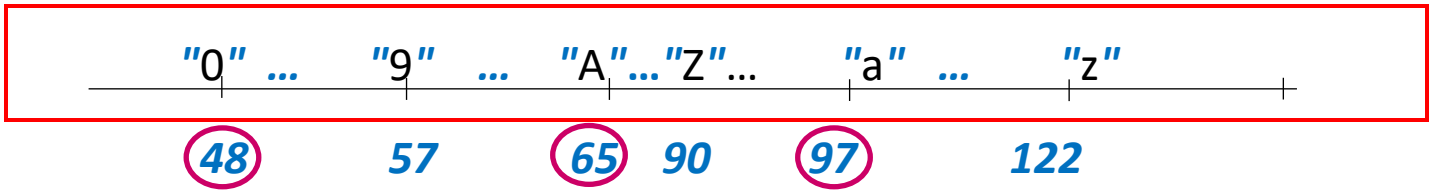
- Une variable de type caractère contient un caractère **et un seul.**
- L'espace " " est le caractère blanc.
- Un caractère doit être placé entre deux guillemets ou entre apostrophes en algorithme et en python.

Déclaration :

Objet	Type/Nature
Nom_variable	caractere

Exemple :

Objet	Type/Nature
c	caractere



À chaque caractère correspond un code appelée code ASCII qui est un entier entre 0 et 255 (voir table des codes ASCII (American Standard Code for Information Interchange).

code	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
10	LF	VT	NP	CR	SO	SI	DLE	DC1	DC2	DC3
20	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
30	RS	US	SP	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	DEL		

Le code ASCII

Opérateur pour les caractères :

Opération	Algorithme	Python	Exemple
Concaténation (la mise bout à bout)	+	+	Ch="a"+"b"ch contient .....
Répétition	*	*	ch= "b"*2 ch contient .....
LES COMPARAISONS **Rq :** La comparaison se fait selon le code ASCII	<><= >= = ≠	<><= >= == !=	'B '< 'b' donne ..... 'A'>='z' donne .....
Appartenance à un intervalle	c ∈['A'..'C']	in	'c' in ['A','b','C'] donne .....

Fonctions prédéfinis sur les caractères :

Algorithme	Python	Rôle	Exemples
n←ord (c)	n=ord(c)	Renvoie le code ASCII du caractère c	n=ord('A') n contient..... n=ord('a') n contient.....
c←chr(n)	c=chr(n)	Renvoie le caractère dont le code ASCII est n	c=chr (65) c contient.....

Remarque :
Python ne supporte pas le type caractère. De là un caractère n'est plus qu'une chaine de caractère de longueur 1
c='A'
t=len(c) # taille de la chaine c
print(c)
donne 1

5. Le type chaîne de caractère (str):

Définition : C'est une succession de n caractère (lettre, symbole, chiffre) avec n ≥ 0.
Une chaîne de caractère doit être placée entre deux guillemets ou entre apostrophes en algorithme et en python

Déclaration :

Objet	Type/Nature
Nom_variable	Chaine de caractère

Exemple :

Objet	Type/Nature
ch	Chaine de caractère

Manipulation de chaîne de caractère :
On peut accéder en lecture et en écriture au ième caractère d'une chaîne Ch en utilisant la notation CH[i] avec 1 ≤ i ≤ Long (Ch).

Exemple Ch='Bonjour'

	B	o	n	j	o	u	r
Indice positif →	0	1	2	3	4	5	6
Ou bien							
Indice negatif→	-7	-6	-5	-4	-3	-2	-1

Remarque :
♣ Si n = 0 alors la chaîne est dite vide ("" : chaîne vide).
♣ Les valeurs chaîne de caractères sont définies entre guillemets
♣ En python les chaînes de caractères sont immuables ((c-à-d **non modifiable** : ni **changement**, ni **suppression** et ni **insertion ni tri**.).

Il est donc, interdit d'écrire par exemple : ch[0]='P'

Exemple:

```
>>> ch="ABCDEFGH"
>>> ch[0]="X" #changer A de l'induce 0 par la valeur "x"
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    ch[0]="X" #changer A de l'induce 0 par la valeur "x"
TypeError: 'str' object does not support item assignment
```

Pour modifier une chaîne de caractères doit construire une nouvelle chaîne qui peut remplacer la précédente avec le même identificateur.

Exemple:

```
>>>
>>> ch="ABCDEFGH"
>>> ch="X"+ch[1:]#changer A de l'indice 0 par la valeur "X"
>>> ch
'XBCDEFGH'
```

Opérateurs :

Opération	Algorithme	Python	Exemple
Concaténation (la mise bout à bout)	+	+	"bon"+"jour" donne ……
Répétition	*	*	ch= "bon"*2 ch contient ……
LES COMPARAISONS **Remarque :** La comparaison se fait 2 à 2 du gauche vers la droite	<><= >= = ≠	<><= >= == !=	'Bonjour '< 'BonJour' donne …… 'Bonjour'<'bonjour' donne ……
Appartenance à un intervalle	c ∈['A'..'C']	in	'c' in ['A','b','C'] donne …… ou 'A<='c'<='C'donne …… Ou 'C' in 'ABC' donne ……

Fonctions prédéfinis sur les chaînes de caractère :

Algorithme	Python	Description	Exemple
l←long(ch)	**l=len(ch)**	Retourne le nombre de caractères de la chaine ch.	ch←"2020" l←long(ch) l contient……
p←pos (ch1, ch2)	**p=ch2.find(ch1)**	Retourne la première position de ch1 dans ch2. Si ch1 n'existe pas dans ch2, retourne -1	ch1←"2" ch2←"2020" ch3←"3" p←pos(ch1,ch2) p contient…… p←pos(ch3,ch2) p contient……
ch←convch(X)	**str(X)**	Retourne la conversion un nombre X en chaine.	ch←convch (2020) ch contient……
a←estnum(ch)	**ch.isdecimal()**	Retourne Vrai si la chaine ch est convertible en numérique, Faux dans le cas contraire.	a←estnum("25") a contient …… b←estnum("a25") a contient ……

n←valeur(ch)	n = int(ch)	Retourne la conversion d'une chaine ch en numérique entière si c'est possible.	ch←"2020" n←valeur (ch) n contient …… ch←"2Info2" n←valeur (ch) ……
	n= float(ch)	Retourne la conversion d'une chaine ch en numérique réelle si c'est possible.	ch←"12.8" n←valeur (ch) n contient ……
ch1←sous_Chaine (ch, d, f)	**ch1=ch[d:f]**	Retourne une partie de la chaine ch à partir de la position djusqu'à la position fexclue.	ch1←souschaine("baccalauréat 2023",5,12) Ch1 contient……
ch1←effacer (ch, d, f)	**ch1=ch[:D]+ch[F:]**	Efface des caractères de la chaine ch à partir de la position d jusqu'à la position f exclue.	ch←"baccalaureat" ch1←effacer(ch, 3,12) ch1 contient ……
ch1←majus(ch)	**ch1=ch.upper()**	Convertit la chaine ch en majuscule.	ch1←majus("Devoir") ch1contient ……

Exercice 1:

Remplir le vide

Opération / méthode	Rôle
len(s)	……
……	Concatène les chaines s1 et s2
……	Donne True si la chaine S contient l'élément "x"
str(n)	……
……	Remplace les minuscules de la chaine S par des majuscules

Exercice 2 :

Soit ch1= "langage de programmation" ch2="45"

Compléter le tableau suivant :

Instruction	valeur	t ype
x←long(ch)	X contient ……	
ch←sous_chaine(ch1,0,7)	Ch1 contient ……	
x←long (sous_chaine(ch1,0,7))	x contient ……	
x←pos ('a', ch1) +valeur(ch2)	x contient ……	
ch←majus((sous_chaine(ch1,0,7)))+ch2	ch contient ……	
b←estnum(ch1)	b contient ……	
b←estnum(ch2)	b contient ……	

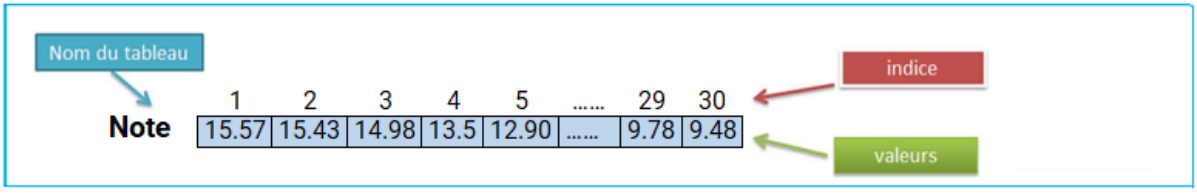
6. Le tableau à une dimension (Vecteur) :

Définition :

Un tableau est une **structure de données** qui permet de ranger **un nombre fini** d'éléments de **même type** désignés par un **identificateur unique**.

Un tableau est caractérisé par :

- son nom (identificateur)
- sa dimension (Le nombre de ses éléments)
- un type de ses éléments.



Déclaration d'une variable de type vecteur :

On déclare un tableau par le mot clé "Tableau", en spécifiant le nombre d'éléments N et leur type de base (Entier, Réel, Caractère, Booléen, Chaine) des éléments du tableau.

En Algorithme	En Python
Tableau de déclaration des objets **Objet** | **Nature / Type** *Ident_tableau* | *Tableau de* ***taille*** *et de* ***type élément***	from numpy import array *Ident_tableau* =array ([***type élément*** ( )] * ***taille***)

Exemple : la déclaration du tableau note pour 30 élèves est la suivante :

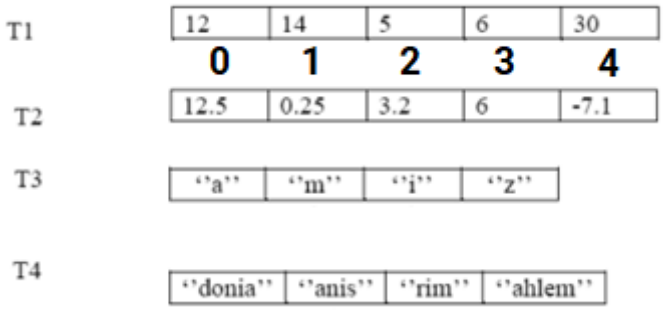
En Algorithme	En Python
Tableau de déclaration des objets **Objet** | **Nature / Type** note | Tableau de 30 réels	from numpy import array note =array ([float ( )] * 30)

Remarque :

♣ Un vecteur est une suite de cases mémoires qui peut contenir des valeurs de même type.
♣ Un vecteur est caractérisé par son nom, sa taille et les types de ses éléments.

Application 1:

Déclarer les 4 tableaux suivants en algorithme et en python



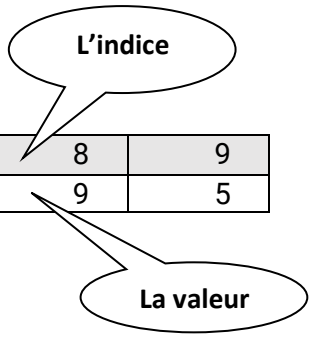
Correction:

En Algorithme	En Python
Tableau de déclaration des objets **Objet** | **Nature / Type** …… | …… …… | …… …… | …… …… | ……	…… …… …… …… ……

Accès aux éléments d'un vecteur :

L'accès à chaque élément se fait par le biais d'un indice.
Pour accéder en lecture ou en écriture au $i^{ième}$ élément d'un tableau, il suffit de spécifier l'identificateur du tableau suivi de l'indice i entre deux crochets

Exemple1 : Soit T un tableau de 10 valeurs d'entières



0	1	2	3	4	5	6	7	8	9
12	76	1	66	55	87	4	8	9	5

T[0] : permet de renvoyer la valeur 12

T[3] : permet de renvoyer la valeur 66

T[8] : permet de renvoyer la valeur 9

Exemple2 : Soit T un tableau de 10 réels

T	10.50	0.25	6.00	-5.00	32.00	569.00	14.00	11.00	43.00	12.00
	0	1	2	3	4	5	6	7	8	9

- 1, 2,…10 : des indices.
- T [0] contient 10.5
- T [1] contient 0.25
- T [9] contient 12.00

Modifier les éléments d'un vecteur :

Pour modifier les éléments d'un tableau, on spécifie le nom du tableau et l'indice de l'élément à modifier entre crochets []. Puis on utilise le signe d'affectation ← en algorithme et le signe d'affectation **=** en python

Syntaxe :

En Algorithme	En Python
Nom_tableau [i]← valeur	Nom_tableau [i]= valeur

Remarque :

Valeur doit être de même type que le tableau **T**

Exemple : Soit T un tableau de 10 valeurs d'entières.

0	1	2	3	4	5	6	7	8	9
12	76	1	66	55	87	4	8	9	5

```
T [0] = 99 // changer la valeur de T [0] par la valeur 99
T [5] = 0 // changer la valeur de T [5] par la valeur 0
T [9] = 10 // changer la valeur de T [0] par la valeur 10
```

0	1	2	3	4	5	6	7	8	9
99	76	1	66	55	0	4	8	9	10

Activité :

Soit T un vecteur de 5 entiers
Donner le contenu de chaque élément du vecteur T après l'exécution de séquence d'instructions suivantes

- T [1] ← 20
- T [2] ← 2
- T [3] ← T [1] DIV T [2]
- T [4] ← T [3] *5
- T [5] ← T [4] + T [3] * T [2]

Solution :

T					
	1	2	3	4	5

Affichage du contenu d'un tableau à une dimension :

L'instruction d'écriture "print()" s'applique aux tableaux comme aux variables plus une boucle répétitive "for"
Pour afficher le contenu d'un tableau
On utilise la structure Pour afin de parcourir le tableau.
//Afficher les éléments du tableau un par un

En Algorithme	En Python
Pour i de Indice Initiale **à** Indice finale **faire** Ecrire(Nom_tableau[i]) **Fin Pour**	for i in range (taille) : print(Nom_tableau[i])

Exemple :

Soit T un tableau de 10 valeurs d'entières.
Pour afficher le tableau T :

En Algorithme	En Python
…… ……	…… ……

Remplissage ou saisie du contenu d'un tableau à une dimension

Pour remplir un tableau on utilise l'instruction de lecture "input ()" qui s'applique aux tableaux comme aux variables plus une boucle répétitive "for …" ou "while…".
//remplir les éléments du tableau un par un

En Algorithme	En Python
Pour i de Indice Initiale **à** Indice finale **faire** Lire(Nom_tableau [i]) **Fin Pour**	**for i in range(taille) :** Nom_tableau [i]=type_element(input(' donner '))

Exemple :

Soit T un tableau de 10 valeurs d'entières.
Pour saisir le tableau T :

En Algorithme	En Python
…… …… ……	…… ……

Application 2 :

Ecrire un script python qui remplit le tableau note avec les notes de 30 élèves puis calcul et affiche la moyenne.

En Algorithme	En Python
Algorithme : …… **Début** …… …… …… …… …… …… …… …… …… …… …… **Fin** **T.D.O : (Tableau de déclaration des objets)** objet / type …… / …… …… / …… …… / ……	…… …… …… …… …… …… …… …… ……

Série 1

Exercice 1 :

Pour chacune des propositions suivantes, mettre dans la case correspondante la lettre **V** si elle est juste ou la lettre **F** si elle est fausse.

1) En algorithme, pour saisir un objet A on doit utiliser l'instruction :

- ☐ écrire(A)
- ☐ saisir(A)
- ☐ lire(A)
- ☐ input(A)

2) En Python, pour afficher un objet S, on peut utiliser l'instruction :

- ☐ print(S)
- ☐ print('S')
- ☐ ecrire(S)
- ☐ print("S=",S)

3) En algorithme, pour affecter la valeur 5 à la variable X, on doit écrire :

- ☐ X = 5
- ☐ X 5
- ☐ X = = 5
- ☐ X := 5

Exercice 2 :

Ecrire un algorithme et une implémentation d'un programme qui permet de convertir une durée de temps donnée T en seconde, en nombre d'heures, minutes, et secondes.

Exemples : T = 5000 le programme affichera : Heures = 1 Minutes = 23 Secondes = 20 T = 6100 le programme affichera : Heures = 1 Minutes = 41 Secondes = 40

Exercice 3

Ecrire un programme qui permet d'afficher la valeur de la surface d'un cercle.

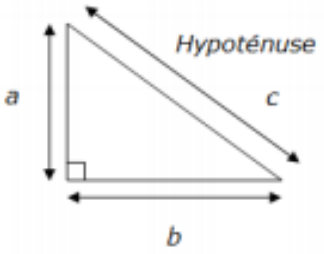
NB : On peut utiliser l'objet pi

Exercice 4

Ecrire un programme qui permet de calculer et d'afficher l'hypoténuse d'un triangle rectangle en appliquant le théorème de Pythagore.

Théorème de Pythagore :

Dans un triangle rectangle, le carré de l'hypoténuse est égal à la somme des carrés des longueurs des côtés de l'angle droit $c^2 = a^2 + b^2$



Exemple :

a = 3.7, b = 6.5 **le programme affichera :** L'hypoténuse est égale à : 7.47

Exercice n°5 : Compléter le tableau suivant

X= random()	Donne au hasard un flottant (réel) dans l'intervalle [0,1[
X=randint(5,10)	……
X=randint(0,6) +12	……

Série 2

Exercice n°1

Pour chacune des propositions suivantes, donner le type de la variable **A**.

Propositions	Type de la variable A
A = input("Saisir A")	
A =float(input("Saisir A"))	
A =int(input("Saisir A"))	

Exercice n°2

Pour chaque instruction donner le résultat et son **type** (**int**, **float, str ou bool**):

Instruction	Résultat	Type
A=3/2		
A=9%3		
A=15//2		
age = 18		
age==17		

Exercice n°3

Question : Mettre Vrai (V) ou Faux (F)

1) Pour affecter a x un entier aleatoire entre 0 et 6 :

☐ x=random() ☐ x=randint(0,6)

2) Soit l'action suivante :

x=randint(2,10)

☐ x dans [0..10] ☐ x dans [2..9] ☐ x dans [2..10]

3)Quelle est la valeur affichée par la séquence d'instructions suivante :

```
from random import *
x= 6 + randint(1,5)
print(x)
```

☐ Valeur aléatoire de type entier comprise entre 6 et 11

☐ Valeur aléatoire de type entier comprise entre 7 et 12

☐ Valeur aléatoire de type entier comprise entre 7 et 11

☐ Valeur aléatoire de type entier comprise entre 6 et 10

4) L'action : print("bonjour")

☐ Lire le message "bonjour" ☐ Saisir le message "bonjour" ☐ Afficher le message "bonjour"

5) Soit l'action : x=17//3+25%3

☐ x=10 ☐ x de type entier ☐ x=6

Exercice n°4

Donner les instructions **Python** permettant de :

Saisir un réel X	……
Afficher le message suivant : « Bon travail pour tous »	……
Affecter à X le reste de la division entière de A par B.	……
Affecter à Z : X^Y	……
Incrémenter (augmenter) la variable B de 1	……

Exercice n°5

Ecrire un algorithme et l'implémentation du programme en python qui permet de :

• Saisir deux entiers m et n.

• Concaténer l'entier m avec l'entier n

• Affecter le résultat de concaténation à une variable p puis afficher le résultat de concaténation. Exemple : m = 167, n =25 le programme affichera : p =16725

Exercice n°6

Ecrire un programme qui permet de calculer et d'afficher le volume d'un cylindre de rayon R et de hauteur H.

Sachant que :

Volume du cylindre = $\pi * R^2 * H$

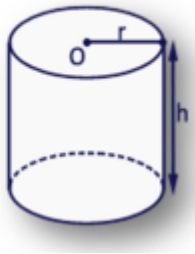
Exemple :

Rayon = 2.7

Hauteur = 10.2

Le programme affichera :

Le volume du cylindre est égal à : 233.60



Série 3

Exercice n°1

Donner l'affichage adéquat pour chaque instruction après l'exécution :

Code	Exécution
`print('donner le nombre \n x')`	…… ……
`s='pa'` `print(s*2)`	……

`s="bonjour tout le monde"`
`x=len(s)`
`print(x)`
	……

`s='bonjour'`
`s=s.upper()`
`print(s)`
	……

Exercice n°2

Donner le résultat de chacune des instructions suivantes :

Instructions	Affichage
ch='radar' x=ch[2]+ch[1]+ch[0] y=ch[2 :] print(x==y)	☐ True ☐ False ☐ "Rad"
ch='yes we can' print(ord(ch[8])	☐ 97 ☐ 99 ☐ 101
mot1='travail' mot2='réussir' c=mot1==mot2 print(type(c))	☐ <Class 'str'> ☐ <Class 'bool'> ☐ <Class 'int'>
mot1='travail' c=chr(len(mot1)+61) print(c)	☐ "C" ☐ "D" ☐ 67

Exercice n°3 :

Soit la chaine suivante ch = "BON courage A TOUS" Donner, en Python, les instructions permettant de : 1) Afficher la longueur de la chaine.

…… 2)

Afficher la position du caractère "C" dans ch.

…… 3)

Convertir la chaine ch en majuscule et l'affecter à ch1.

…… 4)

Modifier le mot "courage" par le mot "chance" dans la chaine ch .

……

Exercice n°4 :

Soit Ch="DevoirdeSynthese"

Déterminer la valeur de chaque objet

Actions	Resultat
x=len(ch) print(x)	…….
y=ch[-1] print(y)	…….
z=ch[:6] print(z)	…….
a=ch.isdecimal() print(a)	…….
b=ch.upper() print(b)	…….
f=ch.find("e") print(f)	…….

Exercice n°5

Ecrire l'algorithme d'un programme intitulé Cryptage qui permet de saisir un mot donné et d'effectuer le chiffrement suivant le principe suivant :

- Permuter le 1er caractère du mot avec le dernier
- Remplacer l'élément du milieu du mot par son code ASCII

Par exemple:
Ch='bonjour'
Résultat =' ron106oub'

Partie 4 : Les structures de contrôle conditionnelles

Définition

La structure de contrôle conditionnelle permet à un programme de choisir et modifier son traitement selon une condition.

On distingue 3 formes de structures conditionnelles :

- Simple
- Généralisées
- À choix multiples

I. La structure de contrôle conditionnelle simple

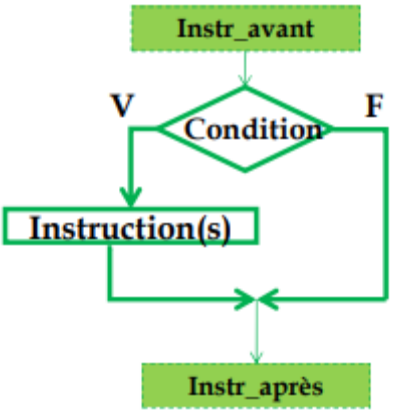
1. Structure conditionnelle simple réduite:

Définition :

La structure conditionnelle simple est dite **structure conditionnelle simple réduite** si on restreint à l'exécution d'un seul traitement et cela dans le cas où la condition est vérifiée.

Syntaxe :

Algorithme	Python
Sicondition **Alors** traitement **Fin si** **Instruction k**	**if** condition**:** Traitement



Remarque :

✓ Cette structure est dite **structure conditionnelle simple réduite.**

✓ Si la condition est vraie alors le traitement sera exécuté sinon, l'exécution passe au bloc d'instructions qui suit la condition.

Activité 1 :

Ecrire un algorithme et son script python qui permet de saisir un entier « N » **strictement positif** et d'afficher sa racine carré.

Algorithme racine
Début

Python :

Fin

Activité 2 :

Ecrire un algorithme et son script python qui permet de saisir un entier « N » et d'afficher sa valeur absolue sans utiliser la fonction abs.

Début Absolue

Python :

Fin

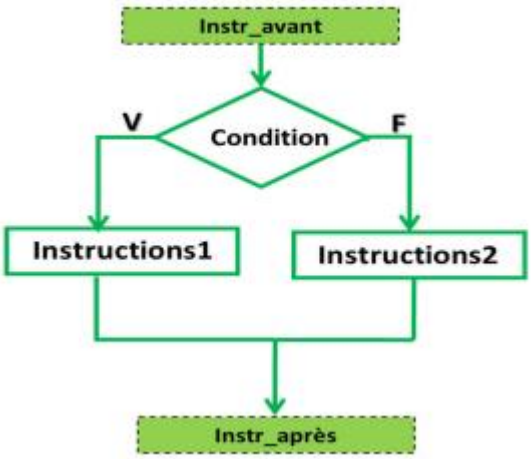
2. La structure de contrôle conditionnelle simple complète ou alternative :

Définition :

La structure de contrôle conditionnelle simple est une structure algorithmique qui fait appel à au maximum deux traitements suivant le résultat de l'évaluation d'une seule condition (vrai / faux).

Syntaxe :

En Algorithme	En Python
Si condition alors traitement1 Si non traitement2 Fin Si	if condition : traitement1 else : traitement2



Remarque :

♣ Lorsque l'évaluation de la condition produit la valeur :
- Vrai : les instructions entre Alors et Fin Si sont exécutées.
- Faux : les instructions entre Alors et Fin Si ne sont pas exécutées.

♣ La condition peut être simple ou composée.

♣ La **condition** est une expression logique qui peut être « vraie» ou « faux »

♣ Lorsque la valeur de la condition est :
- VRAI : le Traitement 1 sera exécuté
- FAUX : le Traitement2 sera exécuté.

♣ Une condition peut être composée de plusieurs expressions logiques liées par les opérateurs booléens.
Exemple : (a>2) ET (b<9) OU (c=3)

♣ On remarque la présence **essentielle** des deux points (**:**)et du **retrait**. Les deux points marquent la fin de la condition. Tout traitement doit faire l'objet d'un léger retrait (indentation ou Tabulation (4 espaces)) indispensable pour être exécuté, sinon une erreur peut se produira.

Activité 1 :

Ecrire un programme qui permet de saisir un entier n et d'afficher leur parité (paire ou impaire)

Solution :

Algorithme	Implémentation en Python
Algorithme……. **Début** …… …… …… …… …… …… …… …… **FinT.D.O**	…… …… …… …… ……

Objet	Type/Nature
…….	…….

Activité 2:

Ecrire un algorithme intitulé **Existence** puis un script python qui permet de déterminer si une chaîne ch1 existe dans une chaîne ch2.

Algorithme	Python
Algorithme : …… **Début** …… …… …… …… …… …… …… …… …… …… **Objet** | **Type/Nature** …… | ……	…… …… …… …… …… …… ……

II. La structure de contrôle généralisée :

Activité 2 :

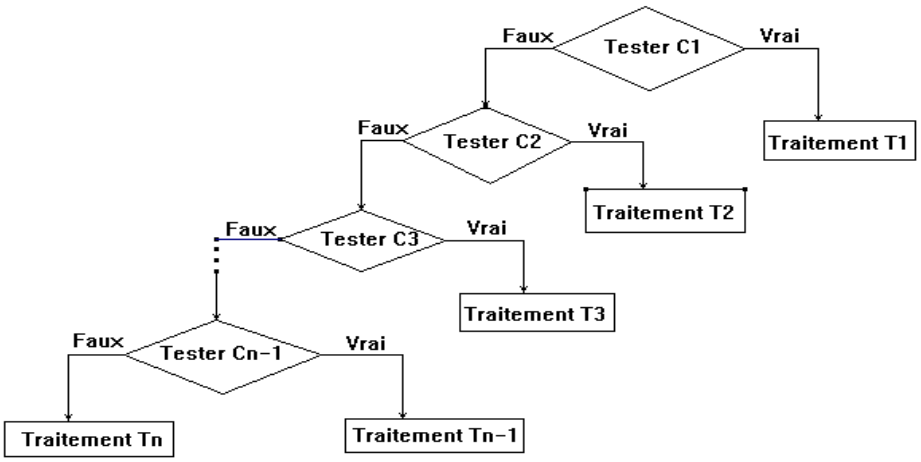
Ecrire un programme qui permet de saisir un entier n et d'afficher leur signe (strictement positif, strictement négatif ou nul).

Exemples :

- ♣ 5 est positif
- ♣ -9 est négatif
- ♣ 0 est nul

Définition :

La forme généralisée est utilisée lorsqu'on a le choix entre plus que deux traitements suivant les résultats des les évaluations des plus que deux conditions.



Syntaxe :

En Algorithme	En Python
Si condition1 alors <traitement1> Si non Si condition2 alors <traitement2> … Si non Si condition n-1 alors <traitement n-1> Sinon <traitement n> Fin Si	if condition1 : <traitement1> elif condition2 : <traitement2> … elif condition N-1 : <traitementn-1> else : <traitementn>

Remarque :

- ♣ L'exécution d'un traitement entraîne automatiquement la non-exécution des autres traitements.
- ♣ La condition peut être simple ou composée.

Solution de l'activité 2 :

Algorithme	Implémentation en Python
Algorithme…… **Début** …… …… …… …… …… …… …… …… …… …… **Fin** **T.D.O : (Tableau de déclaration des objets)** **Objet** | **Type/Nature** ……. | …… …… | ……	…… …… …… …… …… …… …… ……

Activité 3 :

Ecrire un algorithme qui permet de saisir le numéro du mois puis afficher la saison correspondante
Exemple1 : mois=2 saison =hiver
Exemple2 : mois=5 saison =printemps
….

Solution de l'Activité 3

Algorithme	Implémentation en Python
Algorithme …….. **Début** …… …… …… …… …… …… …… …… …… …… …… …… **Fin** **T.D.O : (Tableau de déclaration des objets)** **Objet** | **Type/Nature** …… | …… …… | ……	…… …… …… …… …… …… …… …… …… …… …… ……

III. Structure de choix :

Regardez dans la solution de l'application précédente et remarquez que la structure conditionnelle généralisée est trop longue bien que les conditions portent sur une seule variable, donc on a intérêt de définir une nouvelle structure qui offre une meilleure lisibilité dans ce cas.

Définition :

Cette forme peut remplacer la forme généralisée dans le cas où les conditions portent sur une variable ou une expression de type scalaire.

Vocabulaire et syntaxe :

En Algorithme	En Python
Selon sélecteur **faire** valeur 1 : <traitement 1> valeur 2 : <traitement 2> … Sinon <traitement n> **Fin Selon**	**match** sélecteur : **case** valeur 1 : <traitement 1> **case** valeur 2| valeur2 : <traitement 2> … **case_** : <traitement n>

Remarque :

- Cette structure offre une meilleure lisibilité de la solution que la forme généralisée.
- Dans tous les cas **Sélecteur** doit être de type **scalaire** (pas un réel ou chaîne de caractère)
- En cas **d'égalité du sélecteur** à une valeur (ou fait partie d'une **liste**) l'action qui lui fait associer est exécutée. **Les autres ne seront pas exécutées**
- En cas **d'inégalités** il y aura exécution de la partie **Sinon** (Action n+1) si elle est présente (optionnelle)
- Chaque bloc peut comporter une ou plusieurs instructions.
- La clause sinon et son traitement sont facultatifs dans la structure.

Solution de l'activité 3 avec la structure de choix :

Algorithme	Implémentation en Python
Algorithme……. **Début** …… …… …… …… …… …… …… …… …… …… …… **Fin** **T.D.O : (Tableau de déclaration des objets)** **Objet** | **Type/Nature** …… | …… …… | ……	…… …… …… …… …… …… …… …… …… …… …… …… ……

Application :

Ecrire un algorithme qui permet de saisir un caractère puis afficher sa nature(lettre, chiffre, symbole)

Solution de l'Application

<table>
<tr><th>Algorithme</th><th>Implémentation en Python</th></tr>
<tr><td>
Algorithme …….
Début
……
……
……
……
……
……
……
……
……
……
Fin
T.D.O : (Tableau de déclaration des objets)
<table>
<tr><th>Objet</th><th>Type/Nature</th></tr>
<tr><td>…….</td><td>…….</td></tr>
</table>
</td><td>
……
……
……
……
……
……
……
……
……
……
</td></tr>
</table>

Serie3

Exercice1 :

Cocher ☑ la bonne réponse.

❶	x = int(input("x= ")) R=x **if** x<0 : R=-x print("|" + str(x)+ "| =" + str(R))	X=-9 Le programme affiche : ☐ |-9| =-9 ☐ |-9| =9 ☐ | 9| =9
❷	chaine1= input("Chaîne 1 : ") chaine2= input("Chaîne 2 : ") **if** (**len**(chaine2) >**len**(chaine1))**:** print (chaine2) **else:** print (chaine1)	Chaîne 1 : 2 Ti2 Chaîne 2 : 2 Info2 Le programme affiche : ☐ 2 Ti2 ☐ 2 Info2 ☐ chaine2
❸	A← 7 Si (A ≠ 7) alors C←A*A Sinon C←-A Finsi	Après l'exécution du code suivant, quelle sera la valeur de C 7 -7 49
❹	Quelle est la valeur affichée par la séquence d'instructions suivante : **A← 5** **B←10** **C← 15** **Si (A>B) et (A>C) alors** **Afficher(A)** **Sinon Si B >C alors** **Afficher(B)** **Sinon** **Afficher(C)** **Fin Si**	☐ 5 et 15 ☐ 5 ☐ 15 ☐ 10

Exercice2 :

Ecrire un algorithme intitulé « **Parité** », un TDO et un script python qui permet de saisir un entier « x » et d'afficher le message « entier pair » ou « entier impair ». Un entier pair est divisible par 2.

Exercice3 :

Ecrire un algorithme intitulé « **bissextile** » qui permet de saisir une année (Entier de 4 chiffres) et d'afficher le message « année bissextile » ou « non bissextile ». Une année est dite bissextile si elle est divisible par 4 et non divisible par 100.

Exercice4 :

écrire un **script Python** qui permet de saisir une chaine, un caractère « c », de vérifier et d'afficher un message d'existence de « c » dans « ch ».

Exemple 1: si ch = '' technologie '' et c ='' e ''
alors le prog affichera « e existe dans technologie''
Exemple 2 : si ch = '' technologie '' et c ='' r''
alors le prog affichera « r n'existe pas dans technologie''

Exercice5 :

Ecrire un algorithme puis un script qui permet de saisir successivement un réel (r1), un opérateur (op) et un 2ème réel (r2) puis d'afficher le résultat (R) de l'expression « r1 op r2 » si l'opération est possible SINON affiche les messages convenables en cas d'erreur.

Exemple1 : Pour r1= 50 , op= ''+'' , r2=20 le programme affichera 50.0 +20.0 = 70.00
Exemple2 : Pour r1= 5 , op= ''*'' , r2= 9 le programme affichera 5.0 * 9.0 = 45.00

Partie 5 : Les structures contrôle itératives

Une structure répétitive est aussi appelée boucle nous permettent de gagner énormément de temps en éliminant les répétions. Imaginez que vous avez à exécuter une certaine instruction un certain nombre de fois, disons 100 fois, alors au lieu de taper ces instructions 100 fois, les structures répétitives nous permettent de la taper une seule fois en indiquant le nombre de fois que l'ordinateur doit l'exécuter.

On distingue deux types de boucles :

- Une structure itérative complète où le nombre d'itérations est connu à l'avance : La boucle **Pour … Faire …**
- Une structure itérative à condition d'arrêt où le nombre d'itérations est inconnu à l'avance.
  - La boucle **Répéter … Jusqu'à …**
  - La boucle **Tant que … Faire …**

I. La structure itérative complète : la boucle [Pour … Faire] :

Activité 1 :

Ecrire un programme qui permet d'afficher le mot « Informatique » 1000 fois.

Φ Le nombre de répétition de l'instruction [Ecrire (" Informatique ")] est très grand ce qu'il est impossible d'écrire ces nombres d'instructions donc on a intérêt de définir une nouvelle structure appelé la structure de contrôle itérative complète que nous permet de répéter l'exécution d'une instruction un nombre connu de fois.

Définition :

La structure itérative complète Pour … Faire est utilisée lorsqu'on a un nombre de répétition connu à l'avance d'un traitement donné.

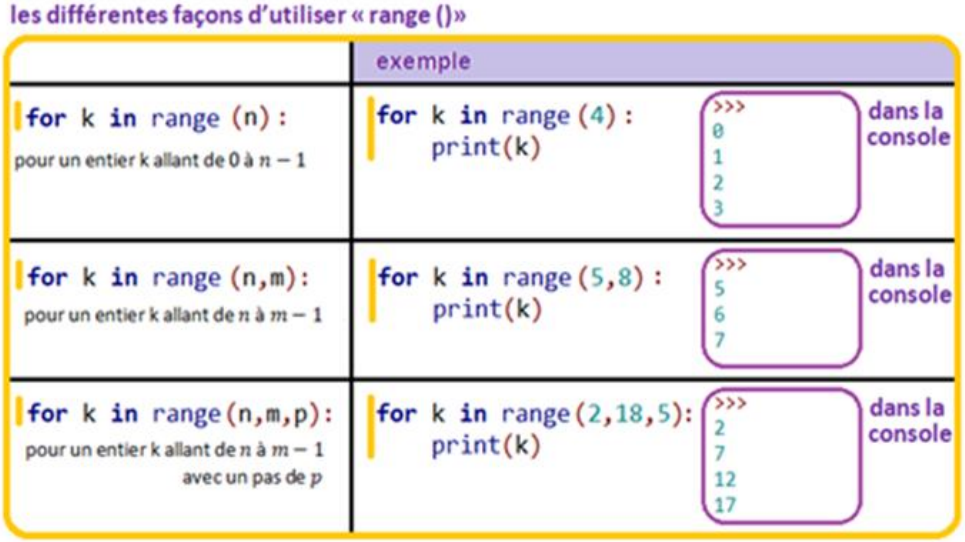
Vocabulaire et syntaxe :

En Algorithme	En Python
Pour <compteur> de <Vi> à <Vf> Faire <traitement> Fin pour	for <compteur> in range (Vi, Vf): <traitement>

Remarque :

- Le compteur est une variable de type scalaire (généralement entier ou caractère) et sa pas d'incrémentation est par défaut de 1.
- Le compteur est initialisé à sa valeur initiale Vi et passe à sa valeur suivante après chaquerépétition jusqu'à attendre la valeur finale Vf.
- Vi et Vf sont de même type ou de type compatible que le compteur.
- Si on utilise range(Vf) alors la valeur initiale Vi =0
- Si on utilise range(Vi, Vf, P) alors sa pas d'incrémentation devient de valeur P
- Dans toutes les structures, chaque traitement peut comporter une ou plusieurs instructions.

les différentes façons d'utiliser « range ()»



Activité 2 :

Ecrire un programme en python dans lequel on affiche la table de multiplication d'un entier donnée

Solution de l'activité 2 :

Algorithme	**Implémentation en Python**
Algorithmemultiplication **Début** …… …… …… …… …… **Fin** **T.D.O :** **Objet** | **Type/Nature** ……. | …….. …… | …….	…… …… ……

Console d'affichage
Saisir un entier : 9 1 * 9 = 9 2 * 9 = 18 3 * 9 = 27 4 * 9 = 36 5 * 9 = 45 6 * 9 = 54 7 * 9 = 63 8 * 9 = 72 9 * 9 = 81 10 * 9 = 90

Exercice 1 :

Ecrire un programme en python dans lequel on affiche les nombres pairs compris entre **20** et **0** sur une seule ligne :

……

……

Exercice 2 :

Ecrire un algorithme d'un programme qui permet d'afficher les caractères d'une chaine.

Algorithme	**Python**
Algorithme …… **Début** …… …… …… …… …… **Fin** **T.D.O :** **Objet** | **Type/Nature** ……. | ……. ……. | …….	…… …… …… Ou bien …… …… ……

Exercice 3 :

Dans chaque cas, dire ce que s'affiche dans la console

```
x=0
for k in range(3,10):
    x=x+1
    print(x)
```

```
x=0
for k in range(3,10):
    print(x)
    x=x+1
```

```
x=0
for k in range(3,10):
    x=x+1
print(x)
```

Exercice 4 :

Ecrire un algorithme d'un programme qui permet d'afficher les caractères d'une chaines.

Algorithme	Python

II. La structure itérative à condition d'arrêt : La boucle [Répéter … Jusqu'à] :

Activité 2 :

Ecrire un algorithme qui demande à l'utilisateur un nombre N compris entre 1 et 3 jusqu'à ce que la réponse convienne. Demander de ressaisir si sa réponse incorrecte.

Définition :

La structure **Répéter … Jusqu'à** est utilisée lorsqu'on a dans le cas où le nombre de répétition d'untraitement donné est inconnu et que le traitement sera exécuté au moins une fois.

S yntaxe :

En Algorithme	En Python
Répéter <traitement> Jusqu'à <condition de sortie>	While Not <condition de sortie> : <traitement>

Remarque :

- La condition d'arrêt est considérée comme une condition de sortie de la boucle car, une fois elle est vérifiée on quitte la boucle.
- La condition de sortie de la boucle peut être simple ou composée.
- La boucle Répéter … Jusqu'à est une structure adaptée pour le contrôle de la saisie des données. Elle impose l'utilisateur d'entrer des données qui vérifient certaines contraintes.
- S'il y a un éventuel compteur, il faut l'initialiser avant la boucle pour assurer son avancement dans la boucle.

Correction Activité 2 :

En Algorithme	En Python
Algorithme controle **Début** …… …… …… …… **Fin** **T.D.O : (Tableau de déclaration des objets)** Objet | Type/Nature n | entier	…… …… ……

Application : Soit l'algorithme suivant :

Algorithme
Algorithme Inconnu **Début** Répéter Ecrire("Saisir n :") lire (n) Jusqu'à (10≤n≤99) C ← n div 10 D ← n mod 10 S ← C + D * 10 Ecrire("S=",S) **Fin**

a) Exécuter manuellement cet algorithme pour les valeurs suivantes :

¬ n=23

n	C	D	S

¬ n=46

n	C	D	S

b) Déduire le rôle de cet algorithme :

.....

III.La structure itérative à condition d'arrêt : La boucle [Tant que … Faire] :

Activité 3 :

On se propose de chercher le PGCD (plus grand commun diviseurs) de deux entiers m et n par la méthode de la différence.

Pour mieux comprendre la méthode, prenons un exemple: si m=10 et n=16

PGCD (10, 16) = PGCD (10, 16-10)
= PGCD (10-6, 6)
= PGCD (4, 6-4)
= PGCD (4-2, 2)
= 2

- Le nombre de répétition est inconnu donc impossible d'opter pour la boucle **Pour … Faire**
- Voyons s'il est possible d'utiliser la boucle **Répéter … Jusqu'à**

Φ Dans le cas ou m=n nous sommes amenés vers une boucle infinie. Dans ce cas il faut que nous n'entrions pas dans la boucle dès que la condition m=n est vérifiée. Donc on a intérêt des définir une nouvelle structure qu'elle peut résoudre ce type de problème.

Définition :

La structure **Tant que … Faire** est utilisée lorsqu'on a dans le cas ou le nombre de répétition d'un traitement donné est inconnu et que le traitement sera exécuté zéro ou un nombre variable de fois.

Syntaxe :

En Algorithme	En Python
Tant que<condition d'entrée>**Faire** <traitement 1> **Fin Tant que**	**while**<condition d'entrée> : <traitement 1>

Remarque :

- La condition d'arrêt est considérée comme une condition d'entrée car, tant qu'elle est Vérifiée on itère encore jusqu'à sa non vérification.
- La condition d'entrée dans la boucle peut être simple ou composée.
- S'il y a un éventuel compteur, il faut l'initialiser avant la boucle pour assurer son avancement dans la boucle.

Solution de l'activité 3 :

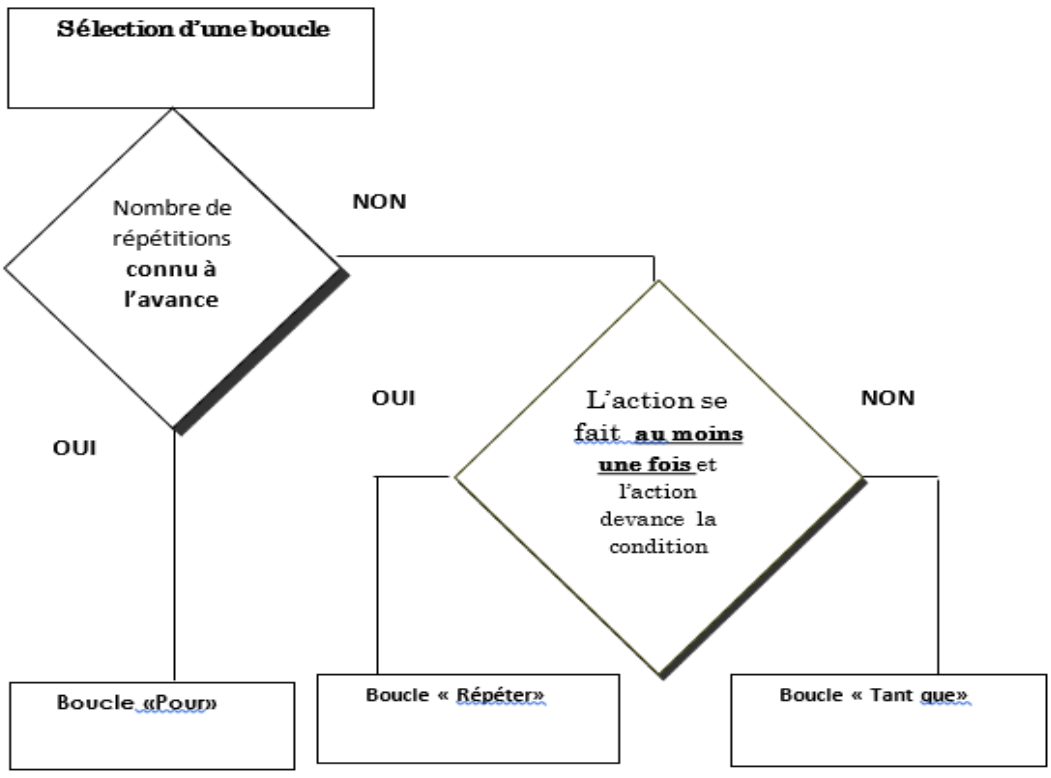
Algorithme……. **Début** …… …… …… …… …… …… …… …… …… **Fin** **T.D.O : (Tableau de déclaration des objets)** **Objet** | **Type/Nature** …… | …… …… | ……	…… …… …… …… …… …… …… ……

Remarques :

La structure **while** réalise un nombre d'itérations inconnu.
le flux d'exécution pour une instruction while :

- Déterminer si la condition est vraie ou fausse ;
- Si elle est fausse, sortir de l'instruction while et poursuivre l'exécution à l'instruction suivante ;
- Si la condition est vraie, exécutez le corps, puis retournez à l'étape 1.
- Si la condition est fausse au départ, le bloc d'instructions ne sera jamais exécuté.
- La condition étant évaluée au début, les variables utilisées dans la condition doivent avoir été initialisées.
- Il faut s'assurer que la condition devienne fausse après un nombre fini d'exécution du traitement (pour éviter une boucle infinie).

SELECTION D'UNE BOUCLE répétitive



Série 5

Exercice1 :
Cocher ☑ la bonne réponse.

❶	x = int(input("x= ")) y= int (input("y= ")) s=1 **for i in range(y+1)** : s= s +x print(s)	X=3 et y= 5 Le programme affiche : ☐ 18 ☐ 19 ☐ 14
❷	x = int(input("x= ")) y= int (input("y= ")) s=0 **for i in range(x,y+1)** : s= s +i print(s)	X=3 et y= 5 Le programme affiche : ☐ 10 ☐ 11 ☐ 12
❸	x=0 y=1 for i in range(1,5): x=x+i y=y*i print('x=',x,'y=',y)	Le programme affiche : ☐ x= 20 y=25 ☐ x= 10 y=24 ☐ x= 10 y=40

Exercice 2 :
☺ Soit le script Python suivant :

```
ch = input ("donner une chaîne de caractères en minuscules")
nv=0
l=len(ch)
for i in range( l ):
    if ch[i] in ['a','u','i','e','o','y'] :
        nv = nv +1
print(nv)
```

1. Exécuter le programme ci-dessus pour chacune des chaînes suivantes:

Chaine ch	"bonjour"	"2 Info2"	"python"	"informatique"
nv	……	……	……	……

2. Déduire le rôle de ce programme : ……
……

Exercice 3
Ecrire un programme qui permet de saisir une chaine de caractère ch non vide puis afficher le nombre d'occurrences d'un caractère donné dans ch.

Exercice 4
Ecrire un algorithme et un programme python intitulé CAL_SOM permettant de calculer la somme des chiffres d'un entier positif saisi au clavier.

Exercice 5 *:*
Ecrire un programme qui calcule la somme des N premiers termes positifs.
Exercice 6 *:*
Ecrire un programme qui calcule la somme des N premiers termes positifs impaires.
Exercice 7 *:*
Ecrire un programme qui calcule la somme des N premiers termes positifs pairs non multiple de 3.
Exercice 8 *:*
Ecrire un programme python intitulé **PARFAIT** qui permet de déterminer si un nombre entier supérieur à 1 est parfait. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs.
Exercice 9 *:*

❶ c← 1 **s← 0** **Répéter** c←c + 1 s← s+ x **Jusqu' à** c >= 7 Afficher (s)	Si x= 5, le programme affiche ☐ 25 ☐ 35 ☐ 30
❷ X=100 Y=50 **while (X != Y):** Y+=10 print(Y) print("X=Y")	

Exercice n° 10:
Ecrire un algorithmequi permet de lire un mot M qui ne dépasse par 15 caractères

Exercice n° 11:
Ecrire un algorithme qui permet de lire un entier de 3 chiffres

Exercice n° 12:
Ecrire un algorithme qui permet de lire un entier de pair

Exercice n°13 :
Ecrire un algorithme intitulé **PALINDROME,** qui permet de lire un mot M qui ne dépasse par 15 caractères et de vérifier s'il est palindrome ou non**. (Un mot palindrome se lit dans les deux sens tel que les mots RADAR, ELLE, …)**.

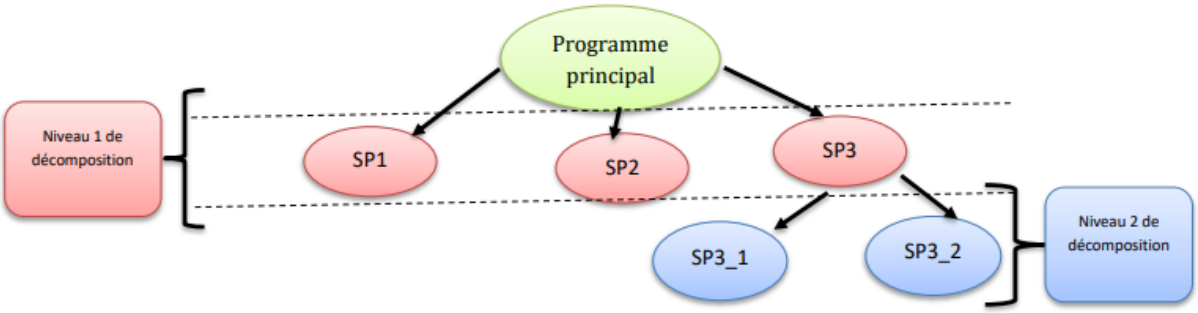
Exercice n°14 :
Ecrire un programme qui permet de saisir une suite d'entier et de calculer leur somme à chaque saisir et afficher cette somme. La liste d'entier se termine par la valeur **-1**.
Exemple **:** Si la suite saisie est 5, 6, 7, -4, -1 la valeur affichée est 14.

Exercice n°15 :
Ecrire un algorithmequi permet de saisir un entier N de 3 chiffres et vérifier s'il est cubique ou non.
Exemple : $153=1^3+5^3+3^3$ est un nombre cubique.

Partie 6 : Les sous programmes

Introduction :

Nous avons vu jusqu'à maintenant les différentes structures nécessaires pour résoudre un tel problème, mais dès que le nombre de traitements augmente le problème devient très complexe et difficile à résoudre. A fin de faciliter la résolution d'un problème complexe et de grande taille, on a intérêt à le décomposer en sous problèmes indépendants et de taille réduite. A chaque sous problème on associe un module assurant sa résolution qu'il peut être une fonction ou une procédure.



--Schéma 1 : *décomposition modulaire*

I. Les fonctions :

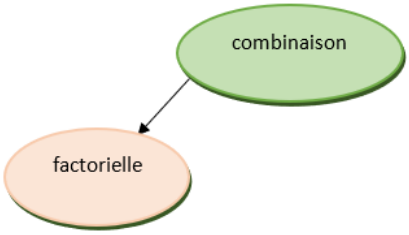
Activité 1 :

Ecrire un algorithme intitulé **Combinaison**et son implémentation en Python d'un programme qui permet de calculer puis d'afficher le nombre de combinaison de P objets

$$C_n^p = \frac{n!}{p! * (n - p)!}$$

N, P sont deux entiers strictement positifs avec N≥P.

Décomposition :

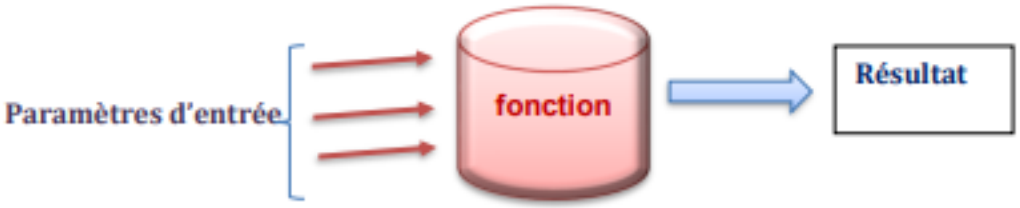


Remarque :

- On constate que le calcul de N !, P ! et (N-P) ! se fait de la même manière et le traitement qui calcule la factorielle se répète trois fois et bien sur le programme dévient très long.
- Donc on a besoin de définir un nouvel outil pour éliminer cette redondance.

Définition :

Une fonction est un sous-programme qui retourne **une seule valeur**de type simple((entier, réel, booléen, caractère, chaîne) générée en fonction des valeurs passées en entrée :



Une fonction qui ne retourne pas de résultat est une procédure

Déclaration :

La déclaration d'une fonction se fait toujours avant le programme appelant :

En Algorithme	En Python
Fonction Nom_fonction (pf1: type1, pf2: type2, … , pfn : typen) : Type_résultat DEBUT <Traitement> Retourner Résultat FIN	def Nom_fonction (pf1, pf2 , … , pfn) : <Traitement> return Résultat

Solution de l'activité 1 :

En Algorithme	En Python
Algorithme de la fonction ……. …… …… …… …… …… …… **Fin** **T.D.O : (Tableau de déclaration des objets locaux)** **Objet** | **Type/Nature** ….. | ……. …… | ……..	…… …… …… …… ……

Remarques :

- Le type de fonction est le type du résultat retourné (Entier, réel, booléen, etc.)
- L'instruction retourne sert à retourner la valeur du résultat
- Une fonction peut avoir de 0 à N paramètres
- Param1, param2, …, sont appelés paramètres (Arguments) : ce sont des variables qui permettent à la fonction de communiquer avec l'extérieur.
- Une fonction est un sous-programme qui retourne une valeur d'un type identique à celui de la fonction.
- Evitez d'y insérer les actions d'entrée et de sortie

Appel d'une fonction :

Une fonction peut être appelée à partir:

❖ du programme principal

❖ d'un autre sous-programme (module) (à condition qu'il soit déclaré à l'intérieur de ce sous-programme ou avant)

L'appel d'une fonction se fait toujours après sa déclaration et elle peut apparaître dans plusieurs emplacements :

Action	En algorithme	En python
affectation	variable←nom_fonction(parE1, …, parEn	Variable=nom_fonction(parE1, …, parEn)
affichage	**écrire**(nom_fonction(parE1, …, parEn))	print(nom_fonction(parE1, …, parEn)
condition	* **Si** (nom_fonction(parE1, …, parEn) = valeur) * **Jusqu'à** (nom_fonction(parE1, …, parEn) = valeur) * **Tant que** (nom_fonction(parE1,…, parEn) = valeur	* **if** (nom_fonction(parE1, …, parEn) == valeur) * **while**(nom_fonction(parE1,…,parEn) == valeur)

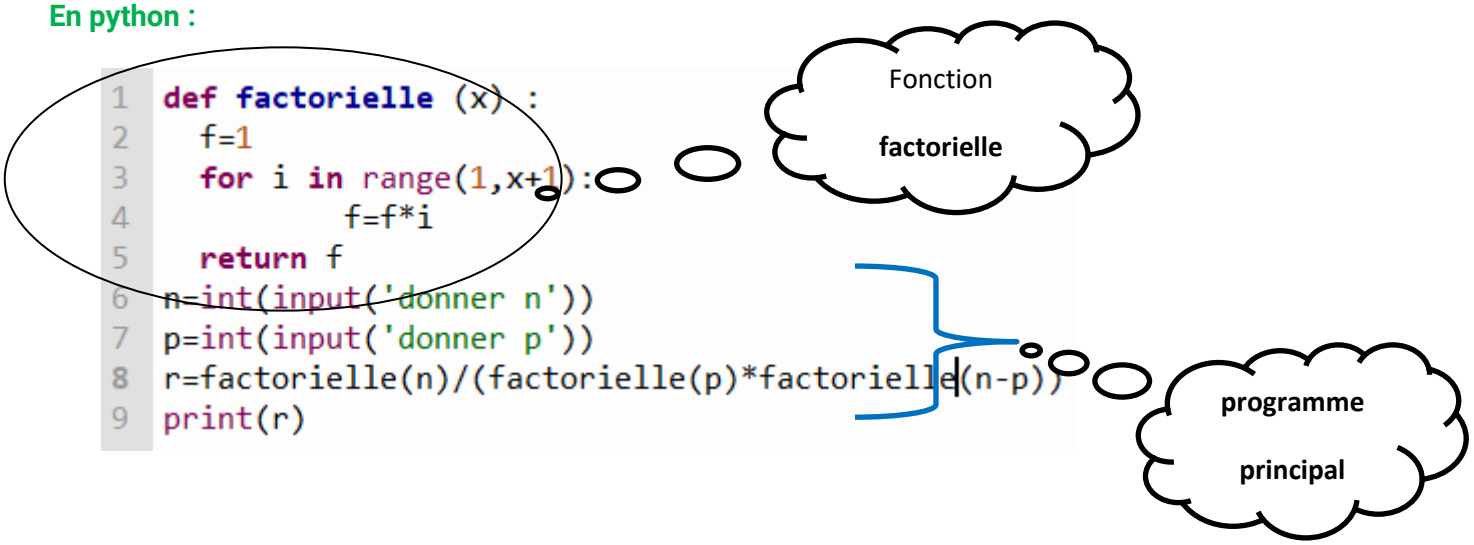
RETENONS:

- Lors de l'appel de la fonction factoriel le paramètre formel x est remplacé par la valeur n et p et n-p (Paramètres effectifs) saisie par l'utilisateur.
- Les paramètres formels doivent s'accorder du point de vue **nombre**, **ordre** et **types** compatibles avec les paramètres effectifs.

Solution de l'activité 1 :

<table>
<tr><th>En algorithme</th><th>En python</th></tr>
<tr><td>
Algorithme du programme principal
Algorithme ……
Début
……
……
……
……
……
Fin
T.D.O globaux
<table>
<tr><th>Objet</th><th>Type</th></tr>
<tr><td>……
……
……</td><td>……
……
……</td></tr>
</table>
</td><td>……
……
……
……</td></tr>
</table>

En python :

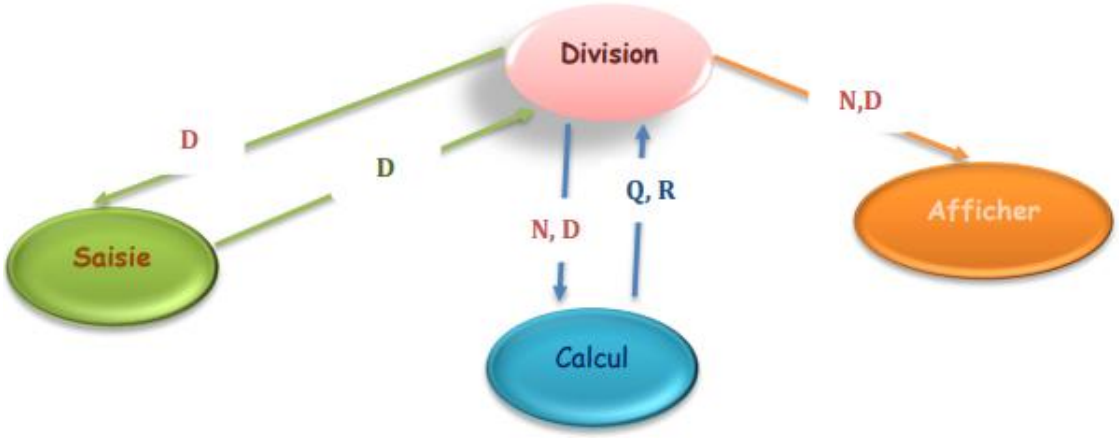


II. Les procédures :

Activité n°2 :

On désire écrire un programme Division qui permet de saisir deux entiers N et D avec D<>0 puis d'afficher leur quotient et reste.

Décomposer le problème en module.

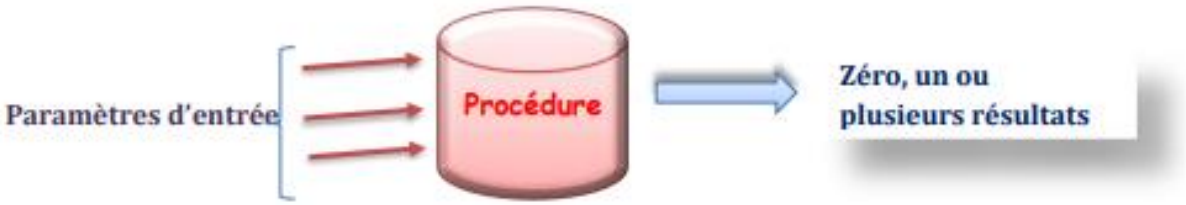


Pour résoudre ce problème on peut utiliser :

- Un module pour la saisie de D et N qui prend deux objets vides puis les remplit,
- Un module pour le calcul de quotient et de reste qui renvoie deux objets,
- Un module affichage qui renvoie zéro objet

Définition :

Une procédure est un sous-programme qui retourne zéro, un ou plusieurs objets en fonction des valeurs passées en entrée :



Déclaration :

En Algorithme	En Python
Procédure Nom_procédure (pf1: type1, pf2: type2, … , pfn: typen) Debut <Traitement> Fin	**def** Nom_procédure (pf1, pf2, … , pfn) : <Traitement> [return resultat]

Remarque :

- En algorithme : Si le mode de passage par référence ou adresse, on ajoutera le symbole @ avant le nom du paramètre
- En python : Eliminer tous les paramètres dont le mode de passage est par référence et les retourner en résultat.

Solution de l'activité 2 :

En Algorithme	En Python
Algorithme de la procédure ……. **……** **Début** …… …… …… …… …… …… **Fin** **Algorithme de la procédure ……** **…….** **début** …… ……. **Fin** **Algorithme de la procédure ……** **……** **début** …… ……. **Fin**	…… …… …… …… …… …… …… …… …… …… …… ……

T.D.O locaux

Objet	Type
……..	……..
……..	……..

Appel d'une fonction

En Algorithme	En Python
nom_procédure (parE1, parE2 , …, parEn)	Dans le cas où la procédure retourne un ou plusieurs résultats : **Variable(s) = Nom_procédure (pe1, pe2, …, pen)** Dans le cas où la procédure ne retourne aucun résultat : **Nom_procédure (pe1, pe2, …, pen)**

N.B : Les paramètres formels et effectifs doivent s'accorder en **nombre, ordre et type.**

Solution :

Algorithme du programme principal **Algorithme Division** **Début** …… …… …… **Fin**	…… …… ……

T.D.O globaux

Objet	Type
……..	……..
……..	……..

Remarque :

- Tout nouvel objet utilisé dans une procédure est appelé objet local.
- Tout objet déclaré dans le programme principal est appelé objet global.
- L'appel d'une procédure peut être effectué au niveau du programme principal ou au niveau d'un module appelant.

Paramètres formels et effectifs :

Activité n°1 :

Ecrire un programme maximum qui permet de saisir deux nombres (x et y) puis d'appeler une fonction max qui retourne la valeur la plus grande.

Solution :

appel de max

Parametres effectifs

```
Algorithme maximum
Début
Ecrire (" Donner le premier : ")
 Lire (x)
 Ecrire (" Donner le deuxième nombre : ")
 Lire (y)
 m←max(x,y)
 Ecrire ("Le maximum de ",x," et ",y," = ",m)
Fin
```

TDO Globaux :

Objet	Type
x,y,m	réel
max	fonction

Declaration de max

```
Fonction max (a,b :réel) :réel
Début
    Si a ≥b alors
        c←a
    sinon
        c←b
    fin si
    retourner c
fin
```

Parametres formels

TDO Locaux :

Objet	Type
c	réel

En python : solution 1

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Le mot-clé **global** permet de **modifier** une variable globale à l'intérieur d'une fonction

En python : solution2

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

a. Les types de paramètres :

Il existe deux types de paramètres :

- **Les paramètres formels :** Ils sont placés dans la déclaration d'un sous-programme, réellement ils n'ont pas de valeurs mais lors de l'appel ils seront restitués par les paramètres effectifs.
  Exemple : **a** et **b** de la fonction **Max**

- **Les paramètres effectifs :** Ils sont placés dans l'appel d'un sous-programme, ils contiennent de valeurs utilisées lors de traitement.

Exemple : **x** et **y** du programme **maximum**

b.Portée des variables :

Selon l'emplacement de déclaration, on peut avoir deux types de variables (objets) dans une fonction : des variables locales ou des variables globales.

La portée désigne l'emplacement de définition et la durée de vie d'une variable (objet).

- **Variable locale :**

Elle est déclarée dans le corps d'un sous-programme. Elle n'est accessible qu'à l'intérieur de module dans lequel a été déclarée.

Exemple : **c** de la fonction **Max**

Exemple 1:

```
Algorithme : Porté_variable1
    procedure afficher()
        Variable X: entier
    Début :
        X←2
        Ecrire("La valeur de X est :",X)
    Fin Procedur
    //Algorithme principal
Début :
        afficher()
        Ecrire(X)
Fin
```

Résultat d'exécution

```
La valeur de X est :2
Erreur x n'est pas définit
```

Dans ce X est un variable locale

Exemple 2:

```
Algorithme : Porté_variable2
    variable X :Entier
    procedure afficher()
        Variable X: entier
    Début :
        X←2
        Ecrire("La valeur de X est :",X)
    Fin Procedur
    //Algorithme principal
Début :
        X←4
        afficher()
        Ecrire(X)
Fin
```

Résultat d'exécution

```
La valeur de X est : 2
X=4
```

X est un variable globale

- **Variable globale :**

Elle est définie en dehors d'un sous-programme. Elle est visible et utilisable dans tout le programme mais la fonction ne peut pas la modifier.

Exemple **: x, y et m** du programme **maximum**

Remarque : en python il est possible de modifier une variable globale dans un module si seulement si cette variable est déclarée dans le sous-programme avec le mot global.

Voir solution 1 programme division

C. Modes de passage (transmission) des paramètres :

Les échanges d'informations entre un module et le programme principal se font par l'intermédiaire des paramètres. Il existe deux principaux types de passages de paramètres qui permettent des usages différents : **Le passage par valeur e**t **le passage par référence (adresse).**

- **Le passage par valeur :**

Dans ce type de passage, le paramètre formel reçoit uniquement une copie de la valeur du paramètre effectif. La valeur de ce dernier ne sera jamais modifiée. Les variables de types numériques et de type non modifiables(chaines de caractères) passent par valeur.

- **Le passage par référence :**

Dans ce type de passage, la fonction utilise l'adresse du paramètre effectif. Lorsqu'on utilise l'adresse du paramètre, on accède directement à son contenu. La valeur de la variable effective sera donc modifiée. Les variables modifiables (tableaux) passent par référence.

Remarque :

En algorithme on ajoutera le symbole @ avant le nom du paramètre dont le mode de passage est par adresse ou référence.

Serie5

Pour chacun des cas suivants donner l'algorithme et le code python d'un sous programme qui permet de :

1) Saisir un caractère Majuscule.
2) Saisir une chaîne de caractère non vide et de longueur maximale égale à 20.
3) Vérifier est-ce qu'une chaîne de caractère donnée est alphabétique ou non.
4) Remplir un tableau T par N entiers positifs croissant.
5) Remplir un tableau T par N caractères Majuscules aléatoires
6) Compter l'occurrence (nombre d'apparition) d'un caractère dans une chaîne.
7) Vérifier la présence d'un caractère dans une chaîne.
8) Déterminer le maximum d'un tableau.
9) Inverser une chaîne de caractère.

10) Soit le programme intitulé **info** qui permet de :

- ✓ Saisir la taille **N** d'un tableau **T,** avec (**1<N<15**).
- ✓ Remplir un tableau **T** par **N** chaînes des caractères tel que la taille de chacune est dans [**3**..**20**].
- ✓ Chercher et afficher tous les chaînes Totalogramme contenue dans **T**.

« Une chaîne de caractères est dite Totalogramme si elle commence et se termine par la même lettre. » (Sans distinction entre majuscule et minuscule)

Exemple : Pour **N=6** :

T	Samir	système	temporairement	Bonjour	ses	elle
	1	**2**	**3**	**4**	**5**	**6**

Les mots totalogramme sont : temporairement, ses, elle

11)Écrire un algorithme et un script python qui permet de trouver le maximum dans un vecteur T de n entiers

Exemple :

Données d'entrée Saisir le nombre d'éléments : n=5

T

3	7	4	9	8

Données de sortie Maximum = 9

Algorithmes de tri et de recherches

Un algorithme de tri est une suite d'instructions servant à ordonner (ranger) une séquence d'éléments de façon croissante ou décroissante.

Exemples :

- Séquence triée : 12 – 15 – 88 – 121 – 122 – 714 – 901 – 2510
- Séquence non triée : 15.5 – 19 – 10 - 201 - 3 – 155

I. Tri à Bulles :

Principe :

Cette méthode consiste à faire remonter le plus grand élément du tableau en comparant les éléments successifs.

1. **Comparer** les éléments du tableau **T** deux à deux,
2. **Permuter** les contenus lorsque **l'ordre n'est pas respecté**,
3. **Refaire** les actions **1** et **2** et ainsi de suite jusqu'à avoir finalement un tableau trié.

Activité :

On désire faire un algorithme d'un programme nommé **classement** qui permet de remplir un tableau T par N entiers quelconques (avec 3<N<10) puis de trier ses élèments par ordre croissant en utilisant le tri à bulles et d'afficher le tableau T avant et après le tri.

Travail demandé :

1- Donner l'algorithme du programme principal

2- Donner les algorithmes des modules envisagés.

procedure saisir ( )

fin saisir

procédure remplir ( )

fin remplir

Procedure Tri ( ) :

Fin Tri

T.D.O.L

objet	type

T.D.O.L

objet	type

Algorithme : **programme_principal**
Début

Fin programme_principal

T.D.O.G

objet	type

Tableau de déclaration de nouveau type

Type	

Python

```
from numpy import *
t=array([int]*20)

def  saisir():
 n=int(input("Donner n : "))
 while n <3 or n>20:
    n=int(input("Donner n :"))
 return n

def  remplir(t,n):
 for i in range (n) :
  t[i]=int(input("Donner t[" +str(i)+ "] : "))

def  afficher(t,n):
 for i in range(n):
    print(t[i],end=" | ")

#fonction de tri par bulles
def tri_bulles(t,n):
    permut = True
    while permut==True :
      permut=False
      for i in range(n-1):
        if t[i]>t[i+1] :
          x=t[i]
          t[i] =t[i+1]
          t[i+1] =x
          permut=True
          # ou bien permutation  t[i],t[i+1]=t[i+1],t[i]


#-----Prog. Principal----
n=saisir()
remplir(t,n)
#appel de fonction
tri_bulles(t,n)
print("tableau final: après le tri")
afficher(t,n)
```

II.La recherche d'un élément dans un tableau :

Introduction :

La recherche d'un élément dans un tableau ou dans une liste de valeur est un traitement très utile en informatique. Parmi les méthodes de recherches, on cite :

- La recherche séquentielle
- La recherche dichotomique.

La recherche séquentielle :

Activité :

Ecrire un programme qui permet de saisir un tableau **t** de **n** entiers n entiers avec (3≤n≤20), puis vérifier si un entier donné **X** existe dans le tableau ou non en utilisant la méthode de recherche séquentielle.

Principe :

La recherche séquentielle est un algorithme qui permet de vérifier l'existence d'un élément dans une série d'éléments. Cette méthode consiste à examiner les éléments de la liste un par un jusqu'à trouver la valeur recherchée (trouve= vrai) ou atteindre la fin du tableau (i = n) Exemple :

Soit un tableau T contenant les dix éléments suivants :

T :	**12**	**10**	**0**	**-5**	**8**	**12**	**-2**	**2**	**40**	**-1**
	1	2	3	4	5	6	7	8	9	10

Pour **X = -2** le programme affichera "**-2 existe dans le tableau**"
Pour **X = 5** le programme affichera "**5 n'existe pas dans le tableau**"

procedure saisir ( )

fin saisir

T.D.O.L

objet	type

procédure remplir ( )

fin remplir

T.D.O.L

objet	type

fonction recherche ( ) :

Fin recherche

Algorithme : **programme_principal**
Début

Fin programme_principal

T.D.O.G

objet	type

Tableau de déclaration de nouveau type

Type	

Partie 8 : Robotique

Robotique

Introduction :

Pourqu'un**système automatisé** réalise**une action**,il faut **une interface** qui fait le lien entre **les capteurs** et **les actionneurs**.

Un ou des capteurs permettent d'acquérir des informations qui sont ensuite traitéespar uneinterface programmable pour piloter un ou des actionneurs qui réalisentl'action (à partir de l'énergie qu'il reçoit) .

Un **capteur** permet de mesurer une grandeur physique (température, obscurité, présence, humidité, lumière, distance, interrupteur, micro..).

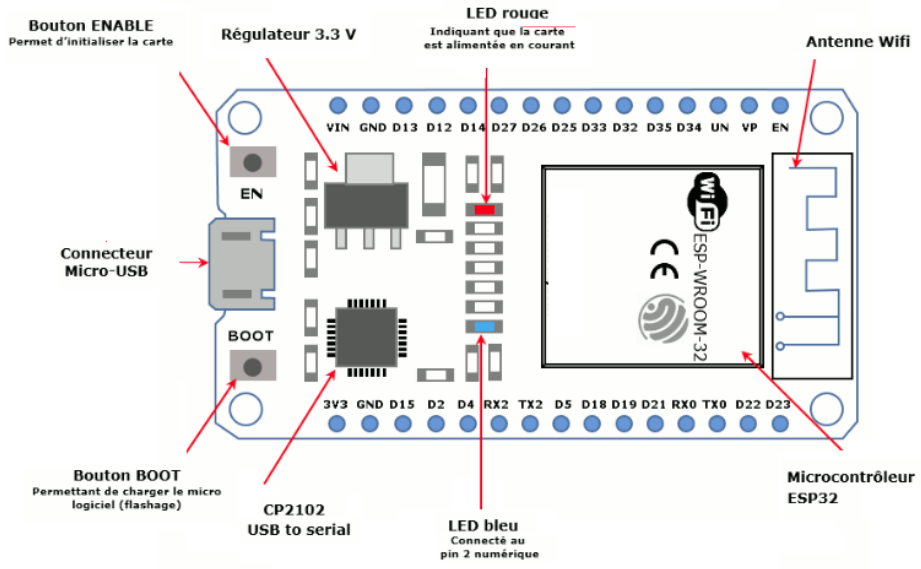
Une **interface** permet la **communication** entre les capteurs et les actionneurs. Elle reçoit les **informations** des capteurs et en fonction d'un **programme,** elle envoie des **ordres** aux actionneurs pour réaliser l'action demandée.

Un **actionneur** convertit une énergie pour réaliser une action(chauffer, éclairer, déplacer...). Exemple : servomoteur, Afficheur, Led, relai, pompe, électrovanne, buzeur, ventilateur

Communication : Infrarouge, Bluetooth, wifi, ondes radio

Prise en main avec la carte ESP32

L'ESP32 développée par la société Espressif , est une carte microcontrôleur de développement à faible coût dédié à l'internet des objets (IoT) et les applications embarquées et doté de communications sans fil Wifi et Bluetooth.

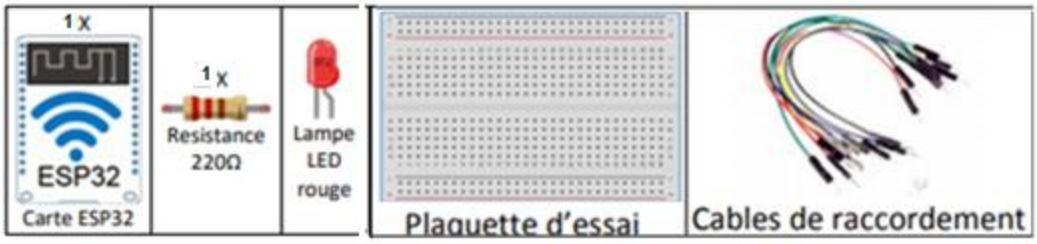


Projet 1 :

Clignoter une diodes LED avec un délai de changement de 1 secondes 5 fois

Pièces nécessaires

Les composantes à utiliser :



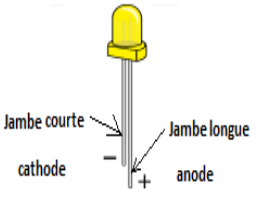
Vous aurez besoin des pièces suivantes :

- **Carte ESP32** :
- **Câblemicro USB**

Pour que ton ordinateur puisse envoyer des instructions à ta carte microcontrôleur, tu vas devoir les connecter avec un câble USB. Si tu souhaites envoyer une instruction comme « allumer la LED » .
Connecte la carte à ton ordinateur en utilisant un câble USB.

- **4 x FILS male-male**
- **3 fils male-femelle**

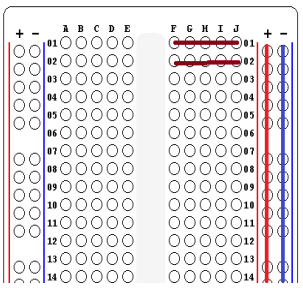
•Led



Sont de petites lumières puissantes utilisées dans de nombreuses applications différente regardez la LED, vous remarquerez que ses jambes sont de longueurs différentes. La long est l'endroit où le courant entre dans la LED. Cette broche doit toujours être connectée à la jambe la plus courte, la «cathode», est la sortie du courant. La jambe courte doit toujours voie d'accès à la terre GND.
Attention au sens d'installation, le méplat indique la cathode

- **Résistance 220 ohms**

une résistance de 220 Ω entre la LED et le GND à éviter un courant excessif qui pourrait brûler la LED.

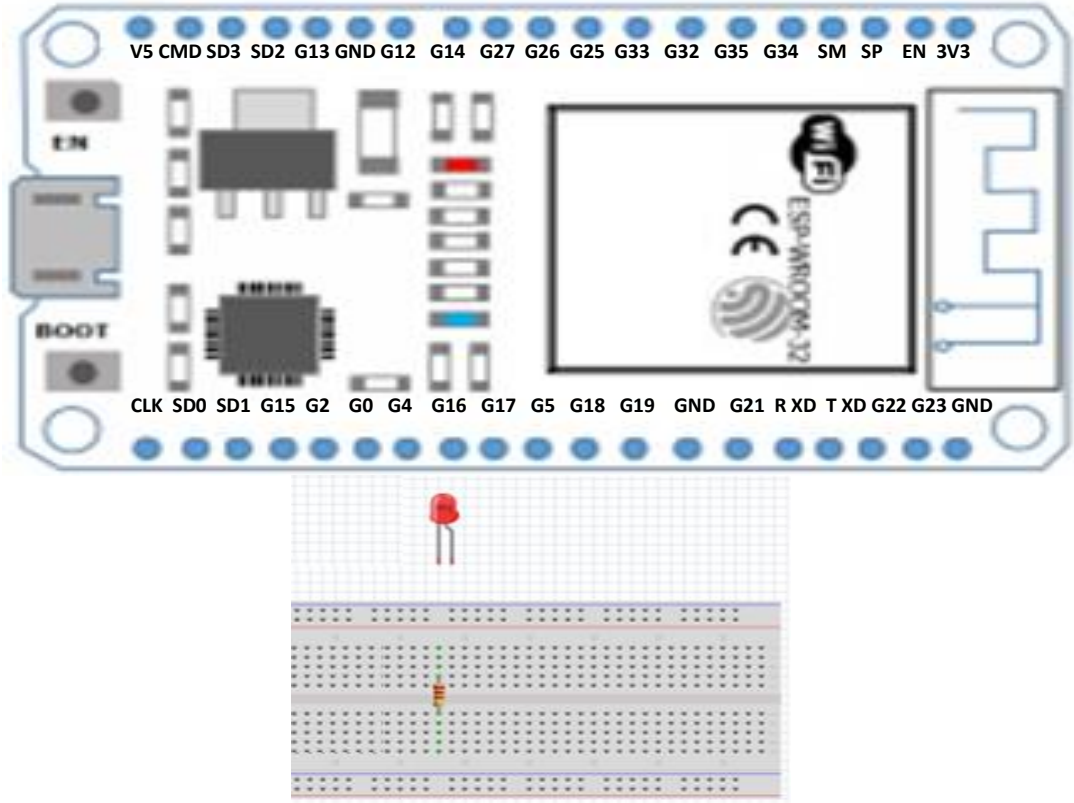


• **La planche d'essai :** Plaquette d'essai (breadboard), permet de réaliser des montages électroniques sans soudure. Les connexions internes sont comme ci-contre.
Les lignes verticales sont les rails d'alimentation. Ceux-ci fonctionnent de la même manière des deux côtés de la planche à pain.
Nous utilisons normalement les bandes rouges pour l'alimentation (3.3 V sur la esp32) et les bleues pour GND.
Les lignes horizontales, chaque groupe

La séquence est la suivante :
• Le feu s'allume pendant 1 secondes
• on attend une seconde
• Le feu s'eteint pendant 1 secondes,
• on attend une seconde
Cette séquence se répète 5 fois

Montage

led	ESP32
-	GND
+	P23



Code :

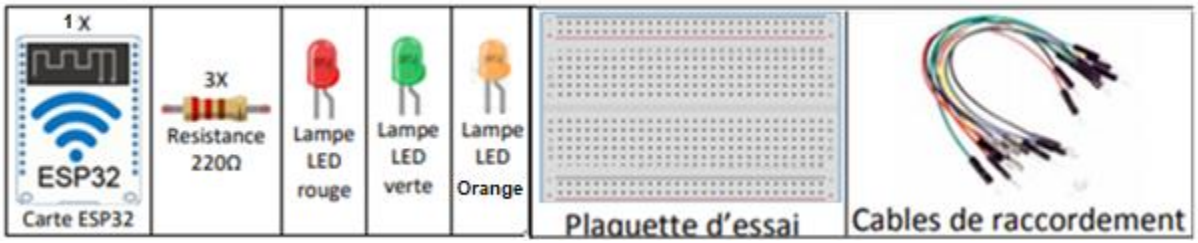
```
from time import sleep
from machine import Pin        # importer pin à partir du module machine
r=Pin(23, Pin.OUT) # Définir le  pin 23 comme output(sortie)
for i in range(5):      # boucle for pour repeter le traitement 5 fois
    r.value(1)
    sleep(1)# Allumer la LED rouge
    r.value(0)
    sleep(1)# Allumer la LED rouge
```

Projet 2 :

Réaliser un feu de circulation en utilisant 3 diodes LED de couleurs (rouge, vert, orange) et 3 résistances de 220 Ω , avec un délai de changement de 5 secondes , commandé par la carte esp32

Pièces nécessaires

Les composantes à utiliser :

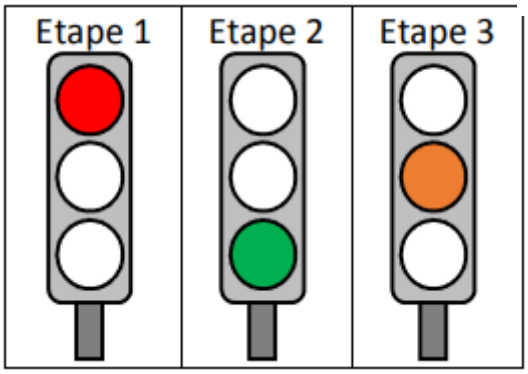


La séquence est la suivante :

• Le feu vert s'allume pendant 3 secondes, puis s'éteint,

• Le feu orange s'allume pendant 1 seconde puis s'éteint et recommence

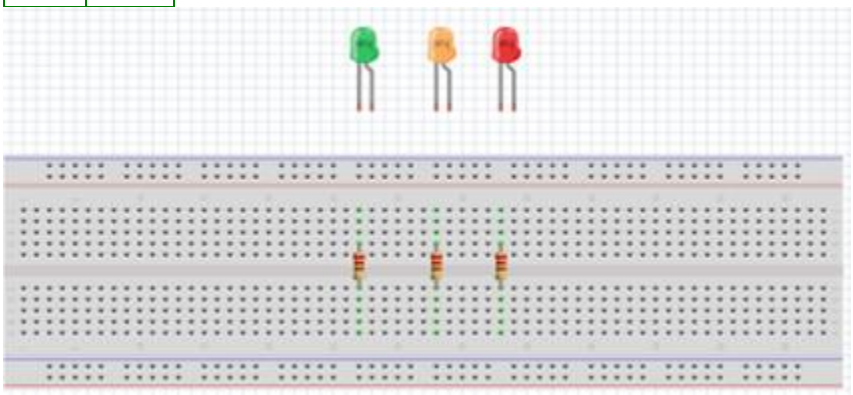
• Le feu rouge s'allume pendant 3 secondes, puis s'éteint,

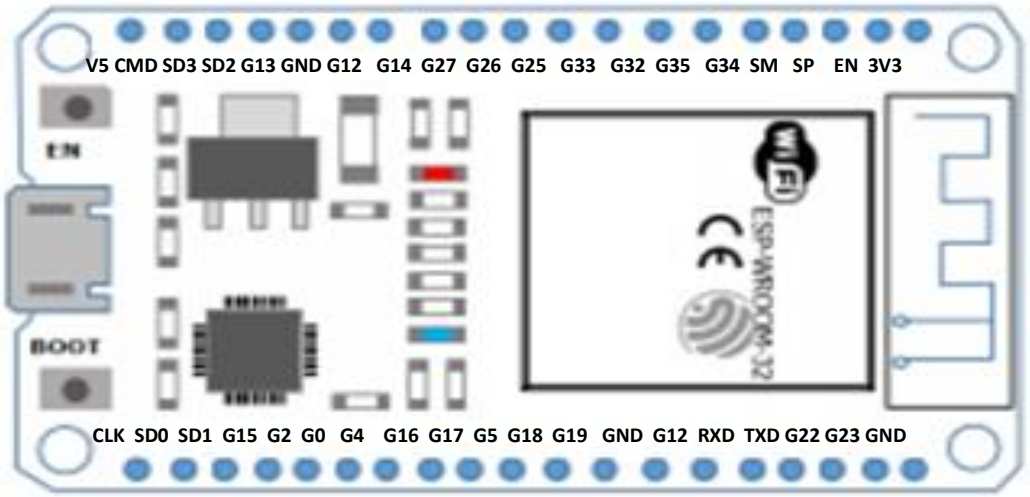


on

Montage

led	ESP32
GND/-	GND
R	P12
O	P14
V	P27





Code :

```
from time import sleep
from machine import Pin        # importer pin à partir du module machine
r=Pin(12, Pin.OUT) # Définir le  pin 12 comme output(sortie)
j=Pin(14, Pin.OUT) # Définir le  pin 14 comme output(sortie)
v=Pin(27, Pin.OUT) # Définir le  pin 27 comme output(sortie)
while True:      # boucle while , boucle infinie puisque condition toujours vrai ->True
    r.value(1)      # Allumer la LED rouge
    j.value(0)
    v.value(0)
    sleep(5)
    r.value(0)      # Allumer la LED jaune
    j.value(1)
    v.value(0)
sleep(5)
    r.value(0)      # Allumer la LED vert
    j.value(0)
    v.value(1)
    sleep(5)
r.value(0)      # Allumer la LED jaune
j.value(1)
    v.value(0)
    sleep(5)
```