Cours, exercices et projets

Version 1.0

3ème * Scientifiques

Sciences Expérimentales – Mathématiques - Techniques

INFORMATIQUE

Nom et Prénom : Classe :

Lycée OTHMAN CHATTI M'SAKEN

2022 - 2023

Présentation du manuel :

Ce manuel est destiné aux élèves du niveau 3ème année secondaire de toutes sections scientifiques que ce soit "Sciences expérimentales", "Mathématiques" ou "Techniques" il est reparti sur deux grands axes :

- **Pensée Computationnelle et Programmation**
- **Internet et nouvelles technologies (Robotique)**

La première partie contient les 5 chapitres suivants :

Chapitre 1 : L'algorithme, les structures de données et les structures simples
Chapitre 2 : Les structures de contrôles conditionnelles
Chapitre 3 : Les structures de contrôles itératives
Chapitre 4 : Le tableau (Vecteur)
Chapitre 5 : Les sous programmes

La deuxième partie contient les 2 chapitres suivants :

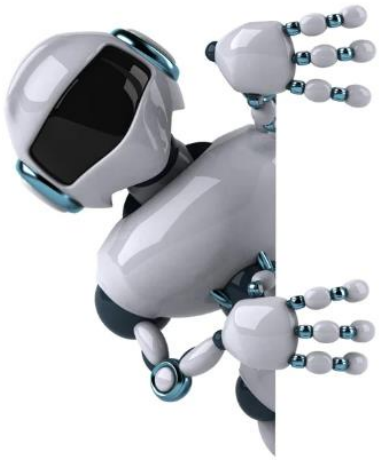
Chapitre 1 : Présentation de la robotique
Chapitre 2 : Pilotage d'un objet connecté

Chaque partie présente une explication simple, détaillée et conviviale des notions algorithmiques dont l'élève aura besoin pour résoudre n'importe quel problème algorithmique à son niveau.
Le manuel contient également une série de 34 exercices et 2 projets robotique pour bien consolider les acquis des élèves durant l'année scolaire.

Je tiens à remercier **Mme Raja Mkhinini** pour sa collaboration dans la préparation de ce manuel.

Pour plus d'information
elbahi.anis@gmail.com

Pensée Computationnelle et Programmation
« Algorithmique et programmation python »



Partie 1 :
L'algorithme, les structures de données et les structures simples

1. L'algorithme :

1.1- Qu'est-ce qu'un algorithme ?

Un **algorithme** est ……
…… permettant de résoudre un problème.

1.2- Squelette d'un algorithme :

Pour écrire un algorithme il faut respecter la syntaxe suivante :

Déclaration des objets

Objet	Type / Nature

Entier
Réel
Booléen
Caractère
Chaine
Tableau

1.3- Qu'est-ce qu'un Langage de programmation ?

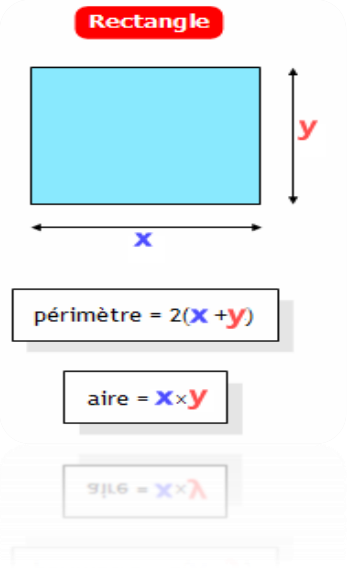
En informatique, un algorithme est la solution d'un problème dans un langage naturel. Cette solution n'est pas …… par l'ordinateur. Pour cela elle doit être traduite en un langage (de programmation) compréhensible à la fois par l'ordinateur et par le programmeur.

la programmation, appelée aussi codage, est l'ensemble des activités qui permettent à un programmeur d'écrire des programmes informatiques en utilisant un langage de programmation.

Il existe beaucoup de langages de programmation voici les plus utilisés :

Exercice 1 (Premier programme)

On désire faire le programme nommé « **rectangle** » qui demande à l'utilisateur **la longueur (x)** et **la largeur (y)** d'un rectangle et calcule son périmètre (**p**) et sa surface (**s**)

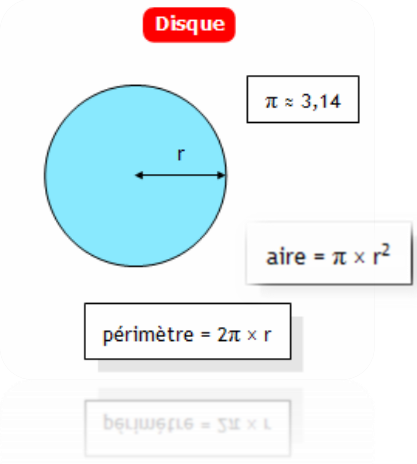


Travail demandé :

1- Faire l'algorithme permettant de résoudre le problème posé.
2- Donner le script python correspondant.

Exercice 2 (Deuxième programme)

Écrire un programme Python qui, à partir de la saisie d'un rayon, calcule et affiche **le périmètre** et **l'aire** d'un cercle.



Constatation :

L'algorithme est une solution permettant de résoudre un problème donné. Pour écrire un algorithme, on peut utiliser des variables, des structures de contrôles (conditionnelles et itératives) , des fonctions prédéfinies et on peut manipuler des opérateurs.

2. Les structures de données :

2.1- Qu'est-ce qu'une variable ?

En programmation, pour résoudre un problème informatique, il faut utiliser des **variables**.
Chaque variable possède un ……, un …… et un ……

Remarque

En algorithmique, on peut utiliser d'autre type d'objets qu'on appelle « constante ».
Une constante est un objet ……
……
……
.

2.2- Les types des variables :

a) type numérique :

Une variable numérique peut être de type …… (int) ou …… (float)

- Une variable de type entier contient une valeur sans virgule : 2, -17, 1254, 0 , -5
- Une variable de type réel contient une valeur avec virgule : 5.2 , -7.0 , 2.55, 15.

a.1) Les opérateurs arithmétiques et de comparaison :

Désignation de l'opération	Priorité des opérateurs	Opérateur en algorithme	Opérateur en python	Exemples
Parenthèses	1	( )	( )	
Multiplication	2	*	*	10 * 2 donne **20**
Division		/	/	10 / 2 donne **5.0**
Division entière		Div	//	10 // 2 donne **5**
Reste de la division entière		Mod	%	10 % 2 donne **0**
Addition	3	+	+	10 + 2 donne **12**
Soustraction		−	−	10 - 2 donne **8**
Égal à	4	=	==	10 == 2 donne **False**
Différent de		≠	!=	10 != 2 donne **True**
Strictement inférieur à		<	<	10 < 2 donne **False**
Strictement supérieur à		>	>	10 > 2 donne **True**
Inférieur ou égal à		⩽	<=	10 <= 2 donne **False**
Supérieur ou égal à		⩾	>=	10 >= 2 donne **True**
L'appartenance (entier, caractère)	5	∈	in	10 in [5,7,10] donne **True**

Exercice 3 (Evaluation d'une expression arithmétique)

1- Pour chaque opérateur donner son équivalent en python et pour chaque expression donner le résultat correspondant.

Opérateur en algorithme	Opérateur en python	Expression	Résultat
+		**5 + 6**	
		"5" + "6"	
-		**4.5 - 2.5**	
/		**11 / 4**	
mod		**11 mod 4**	
div		**11 div 5**	
≠		**"A" ≠ "a"**	
≤		**4 ≤ 4**	
∈		**"A"∈ ["a","A","3"]**	
=		**"ali" = "ALI"**	

2- Compléter le tableau suivant par l'instruction algorithmique, en python ou la valeur de x :

Instruction algorithmique	Instruction en python	Valeur de x	Type de x
x← 15 + 3 * 2 + 5		26	int
x← (18 mod 5) / 2			
	x = (3 % 5) // 2		
x ← abs(-12.5)+3			
x← (15+2*3+4/2+6) div 2			

a.2) Les fonctions prédéfinies sur les types numériques :

Syntaxe Algorithmique	Syntaxe Python	Rôle de la fonction	Exemples
Abs(x)	**abs(x)**	Donne la valeur absolue de x	Abs(-12.5) = 12.5
Ent(x)	**int(x)**	Supprime la partie décimale	Ent(15.63) = 15 Ent(-4.5) = -4
Arrondi(x)	**round(x)**	Donne l'entier le plus proche **NB :** si la partie décimale =0.5 la fonction retourne l'entier pair le plus proche	Arrondi(12.9) = 13 Arrondi(-4.2)=-4 Arrondi(1.5) = 2 Arrondi(2.5)=2
Racine_carrée(x)	**sqrt(x)**	Retourne la racine carrée d'un nombre positif	Racine_carré(16)= 4.0
Aléa(vi,vf)	**randint(vi,vf)**	Donne un entier aléatoire de l'intervalle [vi, vf]	Aléa(10,202) = 15 Aléa(0,1) = 1

Remarque

- En python pour utiliser la fonction sqrt : `from random import randint`
- En python pour utiliser la fonction randint : `from random import randint`

Exercice 4 (Evaluation d'une expression arithmétique / logique)

Compléter le tableau suivant par la valeur ainsi que le type de x (en algorithmique) :

Instruction algorithmique	Valeur de x	Type de x
x← 15 + 3 * 2 + Ent(5.56)		
x← 15. + 3 * 2 + Ent(5.56)		
x←(7+ Arrondi (14.36)) div 2		
x← 20 >10 *1.5		
x← Aléa(10, 20)		
x← Aléa(10, 20) > 30		
x ← arrondi(abs(-12.9)) + Racine_carrée (16)		

b) – Le type booléen :

- Une variable logique ou booléenne (bool) ne peut contenir que la valeur Vrai (True) ou Faux (False)
- La table de vérité du type booléen est la suivante :

x	y	non(x)	x et y	x ou y
Vrai	Vrai	Faux	Vrai	Vrai
Vrai	Faux	Faux	Faux	Vrai
Faux	Vrai	Vrai	Faux	Vrai
Faux	Faux	Vrai	Faux	Faux

Remarque

La priorité des opérateurs logiques :
non (not) > et (and) > ou (or)

Exercice 5 (Evaluation d'une expression logique)

Evaluer les expressions logiques suivantes :

Instruction algorithmique	Valeur de K
K← non ((15+2) = Ent(15.2))	
K← non (10>9) ou (12.59 = Arrondi(12)) et (12 div 2 = 6)	
K← 1*2*3*4*5*6*7*8*0 = 0	
K← ("Anis" = "anis") et (10 = Arrondi(11.58))	

c) – Le type caractère :

c.1) Présentation

- Une variable de type caractère doit contenir un seul caractère.
- Un caractère peut être : "a", "b", …,"A", "B", … ,"0", … ,"9", "*", "=", …
- Chaque caractère possède un code ASCII.

Remarque

Exemples de code ASCII :
"A"=65 "a"=97 "0"=48

c.2) Les fonctions prédéfinies sur le type caractère :

Syntaxe Algorithmique	Syntaxe Python	Rôle de la fonction	Exemples
ord(c)	**ord(c)**	Donne le code ASCII du caractère c	ord("B") = 66 ord("@") = 64 ord("1") = 49
chr(n)	**chr(n)**	Donne le caractère ayant le code ASCII n	chr(48) = "0" chr(65)= "A"

d) Le type chaine de caractères :

d.1) Présentation :

- Une variable de type chaine de caractères peut contenir un ensemble de caractères
- Une chaine vide est une chaine ayant une longueur égale à zéro
- L'indice du premier caractère d'une chaine est zéro.

d.2) Les fonctions prédéfinies sur le type chaine de caractères :

Syntaxe Algorithmique	Syntaxe Python	Rôle de la fonction	Exemples
long(ch)	**len(ch)**	Donne le nombre de caractères de la chaine ch	long("prog") = 4
pos(ch1,ch2)	**ch2.find(ch1)**	Donne la 1ère position de ch1 dans ch2.	pos("o","foot") = 1
convch(d)	**str(d)**	Convertit un nombre décimal en chaine	convch(123)="123"
valeur(ch)	**int(ch)** **float(ch)**	Convertit une chaine en valeur numérique	valeur("12.9") = 12.9
estnum(ch)	**ch.isdigit()**	Donne Vrai si la chaine est convertible en une valeur numérique, sinon elle donne Faux	estnum("2598") = Vrai estnum("25 98") = Faux .
sous_chaine (ch,d,f)	**ch[d :f]**	Retourne une partie de la chaine ch à partir de la position d jusqu'à la position f-1	sous_chaine("Python", 0,4)="Pyth"
efface(ch,d,f)	**ch[:d]+ch[f :]**	Retourne la chaine formée par la partie de la position d jusqu'à la position f-1	efface('Python', 1,4) ="Pon"
majus(ch)	**ch.upper()**	Retourne la chaine en majuscule	majus("Python") = "PYTHON"

Remarque

Pour concaténer (grouper) des chaines de caractères, on peut utiliser l'opérateur **+** :
Par exemple : **"Prog"+"* *"+"python"** donne **"Prog* *python"**

Exercice 6 (Manipulation d'une chaine sous python)

Soit le code Python suivant, donner pour chaque instruction la valeur de k :

```
1  ch1='Programmation'
2  ch2='Bac 2022'
3  k=len(ch1) // 3
4  k= ch1.find('P')
5  k= ch1.find('M')
6  k= ch1.find('m')
7
8  k=ch2[0:3]
9  k=ch2[1:6]
10 k=ch2[:7] +ch1[10:]
11 k= ch2[4:7].isdigit()
12 k=ch2.upper()
```

3. Les structures simples :

Appelées aussi les opérations algorithmiques simples (ou de base) et qui sont :

- L'opération ……
- L'opération ……
- L'opération ……

Remarque

L'opération la plus utilisée en programmation est l'opération ……

Voici la syntaxe des opérations simples :

L'opération d'entrée

Notation Algorithmique	Notation en Python
Ecrire (" Message ") **Lire** (Objet)	*Pour les chaînes de caractères :* Objet = **input** ("Message ") *Pour les entiers :* Objet = **int** (**input** ("Message ")) *Pour les réels :* Objet = **float** (**input** ("Message "))

L'opération de sortie

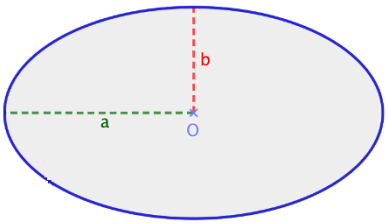
Notation Algorithmique	Notation en Python
Ecrire ("Message", Objet) **Ecrire** ("Message", Expression)	**print** ("Message", Objet, **end** = "") **print** ("Message", Expression, **end** = "")
Ecrire_nl ("Message", Objet) **Ecrire_nl** ("Message", Expression)	**print** ("Message", Objet) **print** ("Message", Expression) ***N.B. :*** *"print" fait un retour à la ligne automatique*

Remarque

la commande **Ecrire** : affiche sans faire un retour à la ligne
la commande **Ecrire_nl** : affiche et fait un retour à la ligne

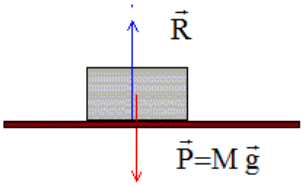
Exercice 7 (Un peu de géométrie)

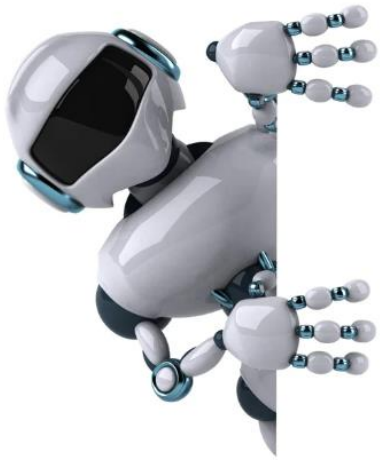
On désire calculer l'aire d'une ellipse en appliquant la formule suivante : **Aire= a*b*π**
On vous demande de faire l'algorithme nommé « ellipse » puis le script python permettant de lire la valeur de a et de celle de b pour calculer et afficher la valeur de l'aire correspondante.



Exercice 8 (Un peu de physique)

Donner l'algorithme qui permet de calculer et d'afficher le poids P d'un solide. Sachant que **P=masse*g** et **g= 9.8**





Partie 2 :

Les structures de contrôles conditionnelles

Les structures conditionnelles :

Les structures conditionnelles permettent de faire ..... parmi plusieurs (et de rendre un programme informatique plus intelligent) . En programmation, on peut distinguer **4 formes** différentes :

Forme 1 : simple réduite	Forme2 : simple complète	Forme 3 : généralisée
Syntaxe algorithmique		
Si Condition **Alors** Traitement **FinSi**	**Si** Condition **Alors** Traitement1 **Sinon** Traitement2 **FinSi**	**Si** Condition1 **Alors** Traitement1 **Sinon Si** Condition2 **Alors** Traitement2 **Sinon Si** Condition3 **Alors** Traitement3 **Sinon** .... **Sinon Si** conditionN-1 **Alors** TraitementN-1 [**Sinon** TraitementN] **FinSi**
En Python		

`if condition_1:`
`  instruction_1.1`
`  instruction_1.2`
`  ...`
	`if condition_1:`
	`  instruction_1.1`
	`  instruction_1.2`
	`  ...`
	`else:`
	`  instruction_n.1`
	`  instruction_n.2`
		`if condition_1:`
		`  instruction_1.1`
		`  instruction_1.2`
		`  ...`
		`elif condition_2:`
		`  instruction_2.1`
		`  instruction_2.2`
		`  ...`
		`...`
		`else:`
		`  instruction_n.1`
		`  instruction_n.2`
		`  ...`

Forme 4 : Structure à choix multiples	
Syntaxe algorithmique	**En python**
Selon <Sélecteur> Valeur1_1[, Valeur1_2, ...] : Traitement1 Valeur2_1 . . Valeur2_2 : Traitement2 ..... [**Sinon** TraitementN] **Fin Selon** **N.B. :** Le sélecteur doit être de type scalaire.	**match** Sélecteur : **case** Val1 : Traitement_1 **case** Val2 | Val3 | Val4 : Traitement_2 **case** Sélecteur if Val5<= Sélecteur <=Val6 : Traitement3 **case** _: Traitement_N

Une structure de contrôle conditionnelle peut être utilisée si on est obligé à tester une condition pour exécuter un traitement.

ATTENTION

En Python (contrairement aux autres langages de programmation) c'est **l'indentation** (les espaces en début de chaque ligne) qui détermine les blocs d'instructions (structures conditionnelles, boucles, etc.).

Exercice 9 (Vérifier la parité)

1- On vous demande de faire un algorithme nommé "**Parité**" qui :
- Permet de saisir un entier x
- Vérifier et afficher s'il est pair ou impair.

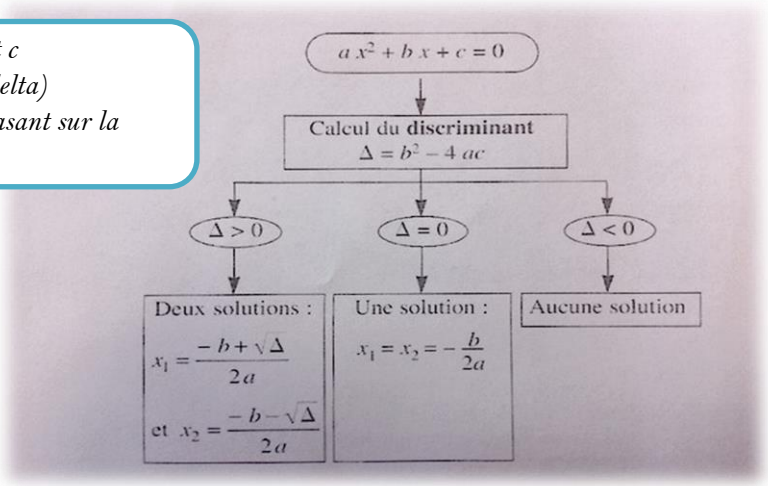
2- Qu'appelle-t-on la structure conditionnelle à utiliser ? → ……

3- Traduire votre algorithme en Python

Exercice 10 (Résolution d'une équation de second degré)

Une équation de second degré s'écrit sous la forme : **$ax^2+bx+c=0$**
Pour résoudre telle équation, il faut suivre la démarche suivante :

1) *Saisir les coefficients a, b et c*
2) *Calculer le discriminant (delta)*
3) *Afficher la solution en se basant sur la valeur du discriminant*



Travail demandé :

Faire le programme Python Permettant de résoudre une équation de second degré.

Exercice 11 (Structure à choix multiple)

On désire faire l'algorithme « **nb_jours** » puis le script python d'un programme qui demande à l'utilisateur le numéro du mois pour lui afficher le nombre du jours correspondant au numéro du mois saisi.
NB : utiliser la structure à choix (selon) pour résoudre le problème.

EXEMPLE

- Si le numéro du mois saisi est 3, le programme doit afficher : Le mois 3 : **31 jours.**
- Si le numéro du mois saisi est 2, le programme doit afficher : Le mois 2 : **28 ou 29 jours.**
- Si le numéro du mois saisi est 19, le programme doit afficher : Le mois 19 : **invalide !**

Exercice 12 (Rappel sur les fonctions prédéfinies)

1. **Soit l'instruction X ← ent (12.33)**
   - Elle permet d'affecter à X la valeur 12 ☐
   - La variable X contiendra une valeur de type entier ☐
   - La variable X contiendra une valeur de type réel ☐

2. **L'instruction R← arrondi (12.75) permet d'effectuer à la variable R**
   - L'entier 12 ☐
   - L'entier 13 ☐
   - Le réel 13.00 ☐

3. **Soit l'instruction C ← sous-chaine ("informatique", 2, 3)**
   - Elle permet d'affecter à C la valeur "**for**" ☐
   - La variable C doit être de type caractère ☐
   - La variable C doit être de type chaine de caractère ☐

4. **L'instruction T←estnum(convch(valeur("3.5")+ long(convch(1.2)))) affecte à T:**
   - 6.5 ☐
   - FAUX ☐
   - VRAI ☐

5. **Soit l'instruction P ← pos ("2", "FIFA 2022")**
   - Elle permet d'affecter à P la valeur 3 ☐
   - Elle permet d'affecter à P la valeur 5 ☐
   - La variable P doit être de type entier ☐

Exercice 13 (Deviner le résultat)

Quel sont les valeurs de A et B après exécution de chaque bloc

```
A ←3
B ←1
Si((A>2) et (B≤1)) alors
        B← 2
Fin si
```

A	B

```
A ←2
A ←0
Si   A=2  alors
     B← 3
Sinon si A=0 alors
     B← 2
Sinon
     B← 1
Fin si
```

A	B

```
A ←2
Si (Non (A<2) ou (A=2)) alors
     B← 3
Sinon
     B← 1
Fin si
```

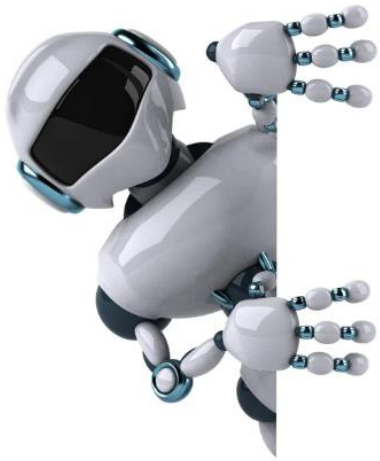
A	B

Exercice 14 (Calculatrice)

a) On désire faire l'algorithme nommé « **calculatrice** » permettant de :
   1- Saisir un réel **x**
   2- Saisir un opérateur (" + ", " - ", " * ", " / ")
   3- Saisir un réel **y**
   4- Afficher le résultat correspondant à l'application de l'opérateur sur les 2 réels x et y.

   NB : pour le cas d'une division par zéro, afficher un message d'erreur.

b) Donner le script Python correspondant



Partie 3 :
Les structures de contrôles itératives

Les boucles

Les structures itératives

Les structures itératives ou structures répétitives ou …… permettent de …… un traitement un certain nombre de fois. En algorithmique on peut distinguer **3 boucles** :

1- La boucle « Pour »:

En Algorithmique

i : ……

Vi : Valeur initiale du compteur

Pour i de Vi à Vf (pas) Faire
Traitement
Fin Pour

Vf : ……

En Python

for i in range (Vi,Vf+1,pas) :
Traitement

EXEMPLE

Soit le bloc suivant :

```
Pour i de 1 à 5 faire
  Ecrire("Prog", i )
Fin Pour
```

1- *Que fait le bloc ?*
2- *Combien est les pas de la boucle ?*
3- *Traduire le bloc en Python*

Remarque

1- La fonction python « range » crée un compteur qui s'incrémente ou se décrémente automatiquement.

Exemple :

- **range(n)** renvoi un itérateur parcourant 0, 1, 2 ... , n − 1 ;
- **range(n,m)** renvoi un itérateur parcourant n, n+1, n+2, ..., m − 1 ;
- **range(n,m,p)** renvoi un itérateur parcourant n, n+p, n+2p , ..., m − 1.

```
>>> A=range(5)
>>> A
range(0, 5)
>>> L=list(A)
>>> L
[0, 1, 2, 3, 4]

>>> list(range(3,8))
[3, 4, 5, 6, 7]

>>> list(range(0,20,5))
[0, 5, 10, 15]
```

Pour afficher le range A, il faut créer une liste

2- Le nombre d'itérations de la boucle « Pour » est : ……..

3- Le **« Pas »** peut être Positif ou négatif, par défaut, le **« Pas »** est égal à ……

4- Le parcours peut être croissant (Début > Fin) ou décroissant (Fin<Début) dans les deux cas, il faut faire attention au **« Pas ».**

5- Voici quelques exemples de boule « Pour » avec python:

```
#afficher les elements d'une liste
L=["Anis", "Ines", "Anas"]
for i in L :
    print(i)
```

```
Anis
Ines
Anas
```

```
#afficher les elements d'une range
for j in range(5) :
    print(j)
```

```
0
1
2
3
4
```

```
#afficher les elements d'une chaine
ch="Anis ELBAHI"
for c in ch :
    print(c)
```

```
A
n
i
s

E
L
B
A
H
I
```

Exercice 15 (La fonction range)

On vous demande de cocher la bonne réponse **(vrai ou faux)** pour chaque instruction et de donner le résultat correct si le résultat proposé est Faux:

Instruction en python	Valeur de i	Vrai	Faux	Résultat correct si Faux
for i in range (5) :	0,1,2,3,4			
for i in range (0,4) :	0,1,2,3,4			
for i in range (2,5) :	2,3,4			
for i in range (5, 2 ,-1) :	5,4,3,2			
for i in range (2,11,3) :	2,5,8			
for i in range (10,-10 ,-5) :	10,5,0,-5			
for i in range (0,len("Python")) :	0,1,2,3,4,5			

2- La boucle « Tant que »:

En Algorithmique	En Python
Tant que (condition(s)) Faire Traitement **Fin tant que**	**while (condition(s)) :** Traitement

Condition(s) de non-arrêt

Tant que la condition est ……, on exécute le traitement de la boucle .
Autrement dit on ne sort de la boucle que si la condition devient ……

EXEMPLE

Soit le bloc suivant :

```
i← 0
Tant que i<5 Faire
   Ecrire("Prog", i )
   i←i+1
Fin tant que
```

1- *Quel est le résultat affiché par le bloc ?*
2- *Traduire le bloc en Python*

ATTENTION La boucle Tant Que

- Le nombre de répétitions n'est pas connu à l'avance
- Le traitement s'exécute au moins zéro fois
- Le traitement s'exécute après le test de la condition de « non-arrêt »
- Si la condition de la boucle « Tant que » est toujours vraie et ne devient jamais fausse, le traitement est répété indéfiniment (à l'infini)

EXEMPLE

Devient en python

```
i←0
tant que (i≠4) Faire
   i← i+1
   Ecrire(" takiacademy " , i )
Fin tant que
```

Ce bloc affiche :
Takiacademy 1
Takiacademy 2
Takiacademy 3
Takiacademy 4

```
i=0
while (i !=4) :
   i=i+1
   print(" takiacademy ", i)
```

Exercice 16 (La boucle while)

Donner le code python pour chaque traitement :
1- En utilisant la boucle while, afficher les entiers de 1 à 100 en utilisant.
2- En utilisant la boucle while saisir un entier n pair.

3- La boucle « Répéter »:

En Algorithmique	En Python
Répéter Traitement Jusqu'à (condition(s))	while (condition(s)) : Traitement

Condition(s) d'arrêt

On exécute le traitement de la boucle Répéter puis on teste la condition : Si la condition est Fausse on répète l'exécution de la boucle et si la condition devient Vraie on « sort » de la boucle

EXEMPLE

Soit le bloc suivant :

```
i← 0
Répéter
   Ecrire("Prog", i )
   i←i+1
Jusqu'à (i≥5)
```

1- *Quel est le résultat affiché par le bloc ?*
2- *Traduire le bloc en Python*

ATTENTION La boucle Répéter

- Le nombre de répétitions n'est pas connu à l'avance
- Le traitement s'exécute au moins une fois
- Le traitement s'exécute avant le test de la condition « d'arrêt »
- N'a pas une correspondance en Python, pour cela elle doit être remplacée par la boucle while .
- Si la condition de la boucle «Répéter» est toujours fausse et ne devient jamais vrai, le traitement est répété indéfiniment (à l'infini)

```
Répéter
   Ecrire("Python" )
Jusqu'à (1>3)
```

EXEMPLE

Devient en python

```
i←0
Répéter
   i← i+1
   Ecrire("takiacademy " , i )
Jusqu'à(i=4)
```

Ce bloc affiche :
Takiacademy 1
Takiacademy 2
Takiacademy 3
Takiacademy 4

```
i=0
while not(i==4) :
   i=i+1
   print("takiacademy ", i)
```

Exercice 17 (La boucle while)

Donner un script python qui permet de saisir une chaine de caractères non vide contenant au maximum 20 caractères .le programme doit par la suite calculer et afficher le nombre de voyelles qui se trouvent dans la chaine ch.

EXEMPLE

Pour la chaine : "Anis ELBAHI"
Le programme doit afficher :
La chaine Anis ELBAHI contient 5 voyelles.

Exercice 18 (Traitement sur les chaines)

On désire faire un programme nommé « **som_chiffres** » qui permet de :

- Saisir une chaine **ch non vide**
- Calculer la somme de ses chiffres.
- Afficher la somme trouvée comme le montre l'exemple suivant :

EXEMPLE

Si l'utilisateur donne la chaine suivante : ch = "**info 1234**"
le programme doit afficher : pour la chaine **info 1234** la somme des chiffres = **10**

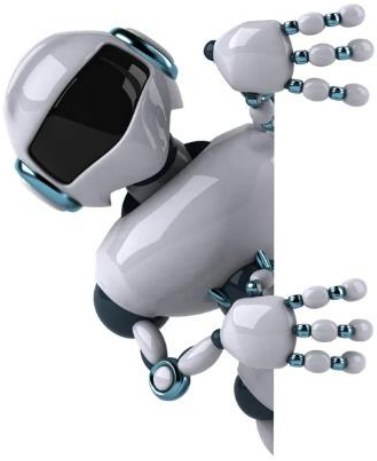
Travail à faire :

1- Faire l'algorithme du programme
2- Déduire le script python correspondant

Exercice 19 (Deviner le résultat)

Cochez la bonne réponse

x=0 **y=1** **for i in range(1,4):** ** x=x+i** ** y=y*i** **print('x=',x,'y=',y)**	Le programme affiche : ☐ x= 6 y=20 ☐ x= 15 y=25 ☐ x= 6 y=6
X←0 **Pour i de 0 à long(ch)-1 faire** ** Si "0"≤ ch[i] ≤ "9" alors** ** X←X+valeur(ch[i])** ** Fin Si** **Fin pour** **Ecrire (x)**	Le programme affiche pour Ch= 'AX3?41R0' ☐ 3410 ☐ 8 ☐ 4
X←0 **Pour i de 0 à long(ch)-1 faire** ** Si "0"≤ ch[i] ≤ "9" alors** ** X←X+1** ** Fin Si** **Fin pour** **Ecrire (x)**	Le programme affiche pour Ch= 'AX3?41R0' ☐ 3410 ☐ 8 ☐ 4
X←"" **Pour i de 0 à long(ch)-1 faire** ** Si "0"≤ ch[i] ≤ "9" alors** ** X←X+ch[i])** ** Fin Si** **Fin pour** **Ecrire (x)**	Le programme affiche pour Ch= 'AX3?41R0' ☐ 3410 ☐ 8 ☐ 4
Pour i de "D" à "A" (pas=-1) faire ** Ecrire(ord(i)-64, i)** **Fin Pour**	Le programme affiche : ☐ A1 B2 C3 D4 ☐ 1A 2B 3C 4C ☐ D4 C3 B2 A1 ☐ 1D 3C 2B 1A
Chx←"" **Pour i de 1 à long(ch)-1 faire** ** Chx←chr(ord(ch[i])+32)** **Fin pour**	Ch←"BONJOUR" Chx contient: ☐ bonjour ☐ Bonjour ☐ BONJOUR ☐ BonjourR



Partie 4 :
Le tableau (Vecteur)

Le tableau :

1- Présentation :

Appelé aussi tableau à une dimension (unidimensionnelle) permet le regroupent d'un ensemble d'éléments de même …….. (algorithmiquement)

T = L'identifiant (nom) du tableau
T contient 6 éléments de type entier

T

15	-199	6	0	17	216
0	1	2	3	4	5

Les indices du tableau

La 5ème case = la case d'indice 4
T[4]=17 : le contenu de la case d'indice 4 = 17

2- Déclaration algorithmique :

Méthode 1

Objet	Type/nature
nom	Tableau de taille et de type élément

Méthode 2

Tableau de déclaration des nouveaux types (TDNT)

Type
Type = Tableau de taille et de type élément

Tableau de déclaration des objets (TDO)

Objet	Type/nature
nom	Type

3- Implémentation en Python :

Pour implémenter un tableau en Python, plusieurs solutions sont possibles :Les listes, les tuples, les tableaux numpy. Dans ce cours nous utiliserons la fonction …… de la bibliothèque …….. de python.

Remarque

Algorithmiquement un tableau doit avoir un nom, une taille et doit contenir des élèments de meme type.

```
#déclaration d'un tableau
from numpy import *
T=array([int()]*5)
```

```
#déclaration d'un tableau
import numpy as np
T=np.array([int()]*5)
```

Exercice 20 (Déclaration d'un tableau)

Soit le vecteur T suivant remplit par 5 réels :

15.5	10.	3.5	5.	10.5
0	1	2	3	4

1- Déclarer le tableau T en algorithmique puis en python.
2- Donner le code python permettant d'afficher le contenu du tableau.

Exercice 21 (Manipulation simple d'un tableau)

1- Donner l'algorithme d'un programme nommé « **tableau_1**» qui permet de faire les tâches suivantes :
   - Saisir N représentant le nombre de cases du tableau (**avec 5≤N≤20**)
   - Remplir un tableau T par N entiers
   - Afficher le contenu du tableau T

2- Donner le script python de votre programme

Exercice 22 (Manipulation avancée d'un tableau)

Donner l'algorithme d'un programme nommé « **Armstrong** » qui permet de :

1- Saisir la taille N du tableau T : **4 ≤ N < 15**
2- Remplir un tableau T par N entiers positifs de 3 chiffres chacun.
3- Afficher tous les entiers d'Armstrong qui se trouvent dans le tableau T.

NB : un entier est dit d'Armstrong s'il est égal à la somme des cubes de ses trois chiffres

Exemple d'entiers d'Armstrong : $153 = 1^3+5^3+3^3$

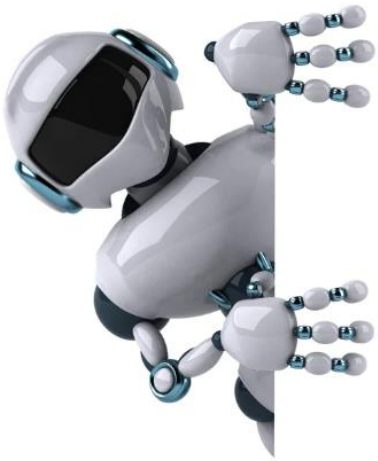
Exercice 23 (Chaine palindrome)

Donner l'algorithme d'un programme nommé « **Palindrome** » qui permet de :

1- Saisir une chaine de caractères non vide.
2- Vérifier et afficher si la chaine saisie est palindrome ou non.

NB : Une chaine palindrome s'il elle peut être lue de droite à gauche comme de gauche à droite.

Exemples de chaines palindromes:
RadaR, ETE, elle, AzizA, A*22*A, …



Partie 5 :
Les sous-programmes (Modularité)

Les sous programmes :

1- Présentation :

En programmation, il est nécessaire de ..... les programmes informatiques en sous programmes indépendants et de difficultés moindres appelés **modules**. Un module peut être ..... ou une ..... (algorithmiquement) .

Les avantages de la décomposition d'un grand problème en sous problèmes :

- *Réduire (décomposer) la difficulté*
- *Partager le travail pour gagner le temps*
- *Plus de clarté, simplicité, ...*

Le tableau suivant présente quelques différences entre une fonction et une procédure.

Sous-programmes En algorithme	
Procédure	**Fonction**
- Retourne ..... - **Exemples de traitements :** ✓ Remplir, afficher, trier un tableau ✓ Saisir un entier, un réel, chaine ✓ Retour de plus qu'une valeur ✓ Affichage de plusieurs résultats ✓ ...	- Retourne ..... ✓ Entier ou Réel ✓ Booléen ✓ Caractère ou chaine - **Exemples de traitements :** ✓ Maximum, minimum, moyenne d'un tableau ✓ Nombre de voyelles, des chiffres dans une chaine ✓ Vérification d'un nombre, d'une chaine, ... ✓ ... Remarque : En algorithmique, une fonction ne retourne jamais un tableau ou plusieurs valeurs en même temps.

2- Les fonctions :

En Algorithmique

Fonction nom_fonction (**pf1** : type, **pf2** :type, ..., **pfn** : type) : **type résultat**
Début
Traitement
Retourner (résultat)
Fin

..... — *Type de résultat retourné par la fonction*

En python

def nom_fonction (**pf1** : type, **pf2** :type, …, **pfn** : type) :
Traitement
return (résultat(s))

.....

Remarque

1- Pour exécuter une fonction, il faut ……..
2- Toutes les instructions qui suivent l'instruction **return** constitue un code mort c'est-à-dire non exécutable par la fonction.
3- Une fonction peut ne pas avoir de paramètres au moment de la définition, dans ce cas il ne faut pas oublier les deux parenthèses ().
4- En python, une fonction peut renvoyer une ou plusieurs valeurs de types différents dans la même instruction return.
5- En algorithmique, une fonction doit retourner …… valeur.

Exercice 24 (Définition et appel d'une fonction en python)

Soit le programme python suivant à compléter :

```
1  def verif(x):
2      if x%2==0:
3          test=True
4      else:
5          test=False
6      return test
7
8
9  y=int(input('donner une entier '))
10
11 k = verif(y)
12
13 if k==True:
14     print(  '.............. ')
15 else:
16     print(  '..............')
```

x est un paramètre formel

Qu'appelle-t-on cette instruction ? ……

y est un paramètre effectif

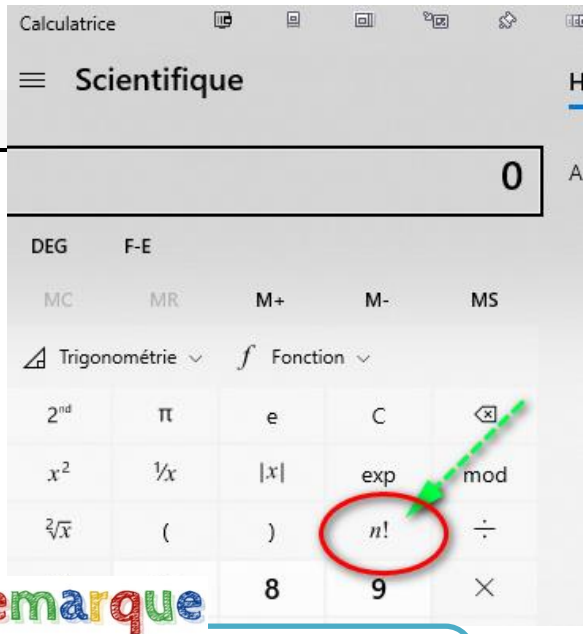
Qu'appelle-t-on cette instruction ? ……

Compléter les pointillés par un message convenable

Exercice 25 (Factorielle)

La factorielle d'un nombre positif N, notée N!
La factorielle se calcule comme suit :
N! = 1*2*3*… * N avec 0!=1

On demande de donner l'algorithme de **la fonction** nommée **fact** qui calcule et renvoie la factorielle d'un entier N passé en paramètre :



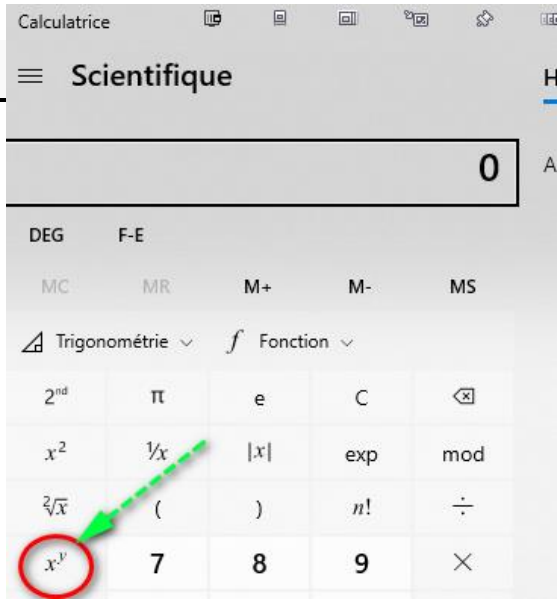
Remarque

La fonction factorial est une fonction prédéfinie dans la bibliothèque math

```
from math import factorial
```

Exercice 26 (Puissance de deux nombres)

On demande de donner l'algorithme de **la fonction** nommée **puis** qui prend en paramètre deux entiers positifs x et y pour calculer et renvoyer la valeur de x à la puissance y (x^y) :



Exercice 27 (Nombre premier)

1- Donner l'algorithme de la fonction nommée premier qui prend en paramètre un entier x et vérifie s'il est premier ou non.
NB : un nombre premier s'il n'a que deux diviseurs distincts 1 et lui-même.
Exemples : 2,3,5,7,11,13,…
2- Traduire la fonction en python

Exercice 28 (Tournage à la main)

1- Soit T un tableau de type TAB rempli par N réels positifs et la fonction Quoi suivante :

```
Fonction Quoi(T :TAB, N :entier) : ……..
Début
  S←0
  Pour i de 0 à N-1 Faire
    S← S+ T[i]
  Fin Pour
  Retourner (S)
Fin
```

2- Compléter par le type de résultat retourné par la fonction.
3- Dresser le TDOL de la fonction Quoi

4- Pour le tableau T suivant quelle est la valeur retournée par la fonction ?

14.	5.5	2.5	3.2	0.8	10.

5- Quel est le rôle de la fonction Quoi ?
6- Donner la traduction python de la fonction Quoi.

Exercice 29 (Bêtisier)

Voici un bêtisier qui génère à chaque fois une erreur, donner la cause de l'erreur dans chaque cas.

`def f()`	
`def fonction () :` `return ('Python')`	

`def puis(x,y):`
`  i=1`
`  p=1`
`  for i in range (1,y+1):`
`  p=p*x`	

`def pair(x):`
`  if k%2 == 0:`
`  return True`
`  else:`
`  return False`	

3- Les procédures :

En Algorithmique

Procédure nom_procédure (**pf1** : type, **pf2** :type, ..., **pfn** : type)
Début
Traitement
Fin

Paramètres formels de la procédure

En python

def nom_procedure (**pf1** : type, **pf2** :type, …, **pfn** : type) :
Traitement

L'entête de la procédure

Remarque

1- Pour exécuter une procédure, il faut ……
2- Une procédure peut retourner 0, 1 ou plusieurs valeurs.
3- En algorithmique, si on veut faire un passage par variable d'un paramètre formel il faut le précéder par « @ » dans l'entête de la procédure.

Exercice 30 (Problème 1 – corrigé)

On désire faire un programme intitulé "**Tableau**" qui permet de faire les tâches suivantes :

- Saisir N le nombre de cases du tableau à remplir sachant que 5≤N≤20
- Remplir le tableau T par N entiers quelconques
- Afficher le contenu du tableau T
- Chercher et afficher le maximum du tableau T

Travail à faire :
1- Donner l'algorithme du programme principal
2- Donner les algorithmes des modules envisagés

1) L'algorithme du programme principal :

```
ALGORITHME Tableau
DEBUT
   Saisir(n)
   Remplir(T,n)
   Afficher(T,n)
   Mx←Maximum(T,n)
   Ecrire (" Le maximum du tableau : " , Mx)
FIN
```

TDNT:

Types
TAB = tableau de 20 entiers

TDOG:

Objet	Type / Nature
n	Entier
T	TAB
Mx	Réel
Saisir	Procédure
Remplir	Procédure
Afficher	Procédure
Maximum	Fonction

1) Les algorithmes des modules envisagés :

```
Procédure Saisir (@n :entier)
Début
  Ecrire ("donner N ") , Lire (N)
  Tant que (Non (5≤N≤20)) Faire
     Ecrire ("donner N ") , Lire (N)
  Fin Tant que
Fin
```

Ou bien

```
Procédure Saisir (@n :entier)
Début
  Répéter
     Ecrire ("donner N ") , Lire (N)
  Jusqu'à (5≤N≤20)
Fin
```

```
Procédure Saisir (@T : TAB , n :entier)
Début
   Pour i de 0 à N-1 faire
      Ecrire ("donner un entier ") , Lire (T[i])
   Fin Pour
Fin
```

TDOL :

Objet	Type / Nature
i	Entier

```
Procédure Saisir (T : TAB , n :entier)
Début
   Ecrire("le contenu du tableau T : ")
   Pour i de 0 à N-1 faire
     Ecrire_nl (T[i])
   Fin Pour
Fin
```

TDOL :

Objet	Type / Nature
i	Entier

```
Fonction Maximum (T :TAB , n :entier) : entier
Début
   Mx←T[0]
   Pour i de 1 à N-1 faire
      Si (T[i]>Mx) alors
         Mx←T[i]
      Fin Si
   Fin Pour
   Retourner (Mx)
Fin
```

TDOL :

Objet	Type / Nature
i, Mx	Entier

Remarque

*La commande **retourner** doit être utilisée avec les fonctions.*
En algorithmique, une fonction doit retourner une seule valeur de type simple
*En algorithmique il ne faut pas oublier de mentionner **le type de résultat** retourné par la fonction*

4- Les variables locales et globales:

Variable locale :
Déclarée dans un sous-programme et n'est visible (utilisable) qu'à l'intérieur de de celui-ci.
Variable globale :
Déclarée en dehors d'un sous-programme et visible (utilisable) dans tous le programme.

Exercice 31 (Visibilité des variables)

Soit le programme python suivant :

```
1  def module1():
2      a=2
3      b=6
4      print(a+b+c)
5
6  def module2():
7      x=a*2
8      return(x)
9
10 c=5
11 module1()
12 print(module2())
```

Questions :

1- Quelles sont le(s) variable(s) globale(s) du programme ?
……

2- Quelles sont le(s) variable(s) locale(s) du module1 ?
……

3- Quelles sont le(s) variable(s) locale(s) du module2 ?
……

4- Pourquoi le programme affiche le message d'erreur :
`NameError: name 'a' is not defined`
dans la ligne 7
……
……
……

5- Paramètres formels et effectifs:

Paramètre formel : Se trouve dans …… d'un sous-programme.
Paramètre effectif : Se trouve dans …… d'un sous-programme.

Remarque

Les paramètres formels et effectifs doivent s'accorder : 1- …… 2- …… 3- ……
Les paramètres formels et effectifs peuvent avoir des noms différents mais il vaut mieux avoir les mêmes noms si possibles.

6- Mode de passage des paramètres:

Passage par valeur :
la valeur du paramètre effectif ne sera jamais modifiée après l'exécution d'un sous-programme.

Passage par adresse (par référence) :
la valeur du paramètre effectif sera modifiée après l'exécution d'un sous-programme .

Remarque

En algorithme, si le mode de passage est par adresse, on ajoutera le symbole @ avant le nom du paramètre.

Exercice 32 (Problème 2)

On désire faire un programme nommé «**problème** » qui permet de faire les tâches suivantes :

1- **Saisir** un entier n (**avec 2<n<10**).
2- **Remplir** un tableau t par n entiers.
3- **Afficher** le tableau t après le remplissage.
4- **Calculer** et afficher la moyenne du tableau t.
5- **Chercher** et afficher le maximum du tableau t.

NB: il faut décomposer votre programme en modules

Travail à faire :

1- Donner l'algorithme du programme principal
2- Donner les algorithmes des modules envisagés
3- Faire le programme python correspondant

Exercice 33 (Tournage à la main)

Soit l'algorithme de la fonction "travail" suivant :

0) Fonction Travail (N : entier) : ……..
1) R ← 0
2) Répéter
R ← R + N mod 10
N ← N div 10
Jusqu'à (N=0)
3) Retourner (R)
4) Fin travail

Remarque

On remarque que l'algorithme peut avoir une structure différente à celle que nous utilisons, mais le principe reste le même

Travail à faire :

1- Compléter par le type de résultat retourné par la fonction travail.
2- Compléter le Tableau de Déclaration des Objets Locaux de la fonction Travail.

Objet	Type / Nature
……	……

3- Quelle est la valeur retournée par la fonction Travail pour N = 125
Pour N = 125 le résultat = ……
4- Déduire le rôle de la fonction travail.

Exercice 34 (Formatage de l'affichage)

Ecrire un programme qui permet de saisir une chaine **ch** *non vide de taille maximale égale à 10* formée uniquement par des lettres alphabétiques majuscules et de l'afficher sous la forme d'un triangle comme le montre l'exemple suivant.

EXEMPLE

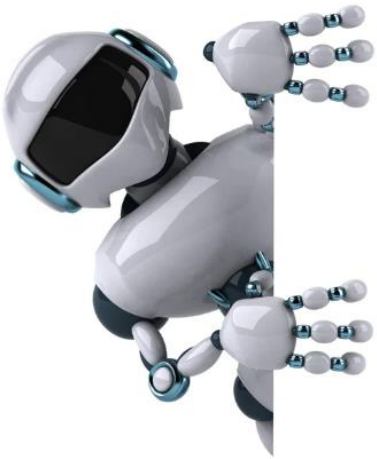
Pour **ch** =''ELBAHI'', on aura :

E
EL
ELB
ELBA
ELBAH
ELBAHI

Travail à faire :

1- Donner l'algorithme du programme principal
2- Faire les algorithmes des modules proposés.
3- Donner la traduction Python de votre programme

Internet et nouvelles technologies
« la robotique »



Partie 1 :
Présentation de la robotique

La robotique :

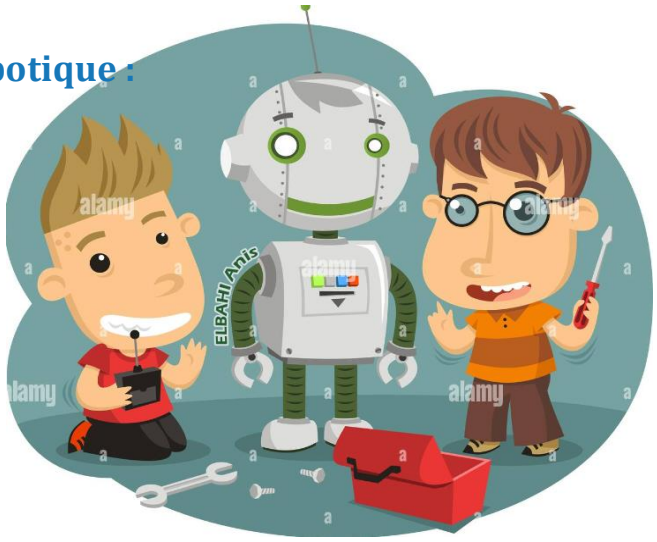
1- Présentation :

La robotique est l'ensemble des domaines scientifiques et industriels en rapport avec la conception, la fabrication et la programmation des robots.

2- Domaines d'application de la robotique :

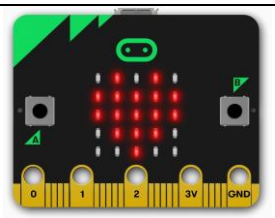
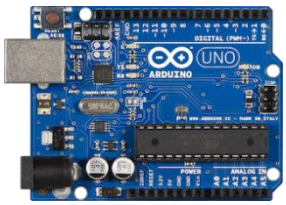
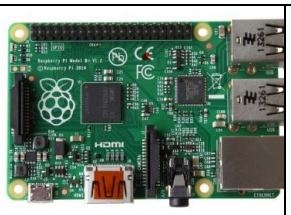
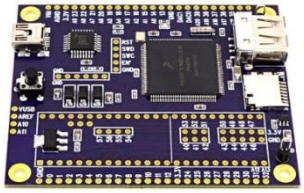
Les robots sont à peu près partout.

- ……
- ……
- ……
- ……
- ……
- ……



3- Enseignement de la robotique :

Pour enseigner la robotique , on peut utiliser des …… dédiées à la robotique comme :

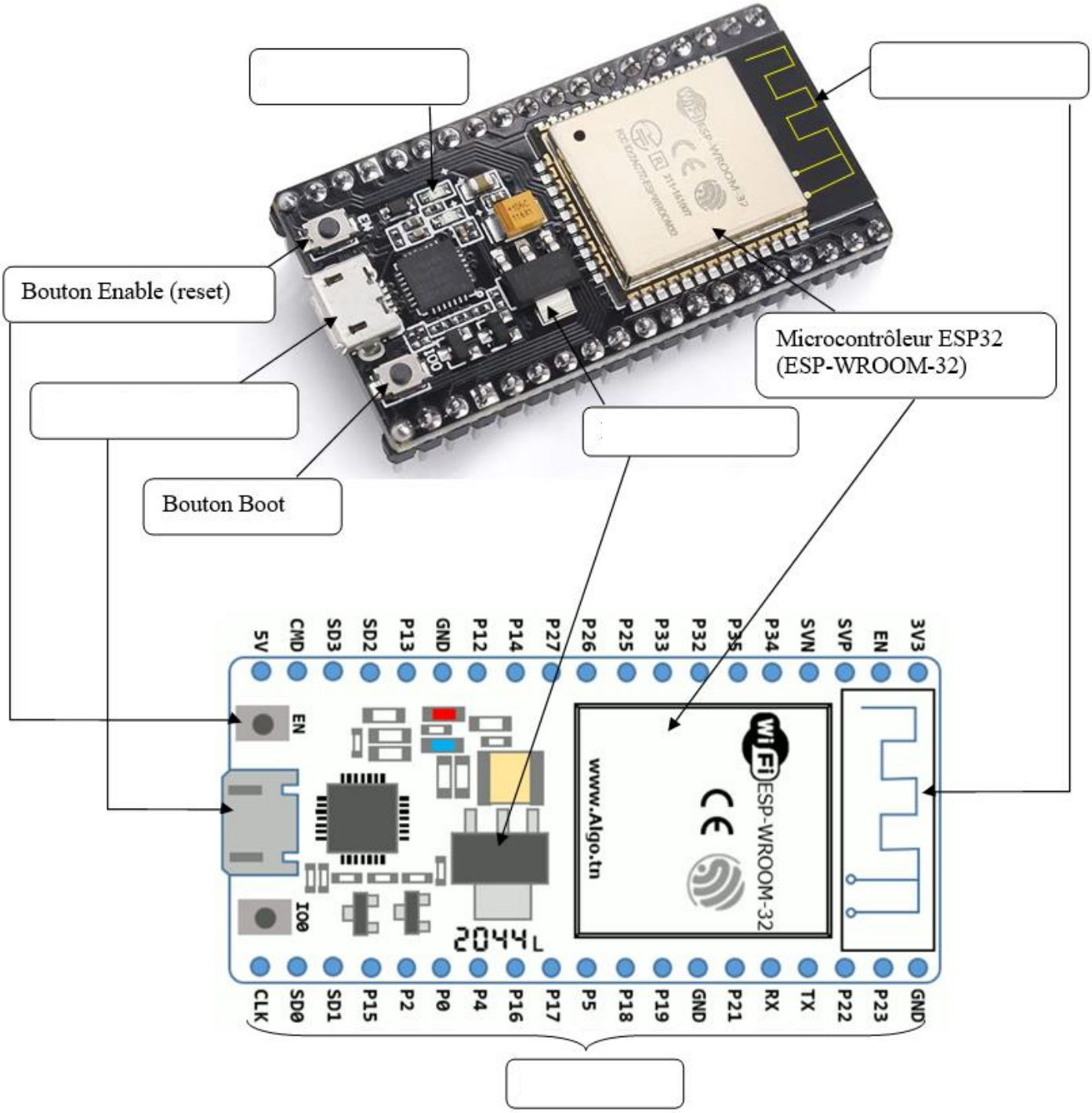
			
La carte Micr- bit	**La carte Arduino**	**La carte Raspberry**	**La carte Lattepanda**
			
La carte Pycom	**La carte teensy**	**La carte Mcore**	

Dans ce cours on va apprendre la robotique à l'aide de la carte ESP32.

4- la carte ESP32 :

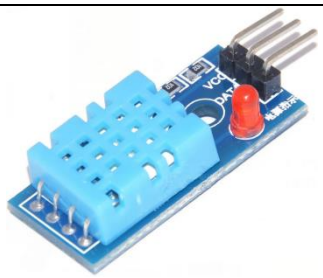
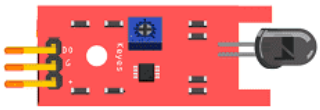
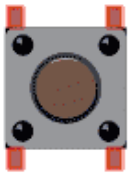
4.1 - Présentation de la carte :

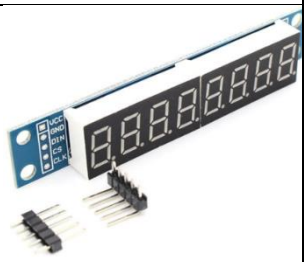
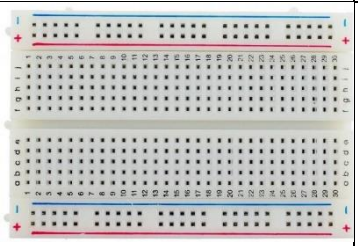
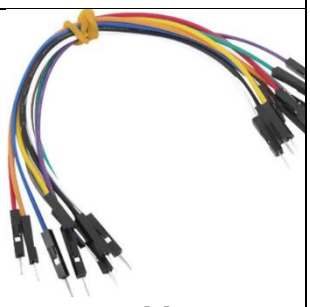
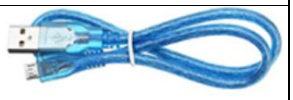
La carte ESP32 est une petite …… développée par la société Espressif Systems.



4.2- Accessoires de la carte ESP2 :

Plusieurs composantes peuvent être connectés à la carte ESP32 voici quelques exemples :

<table>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
Capteur de température et d'humidité</td>
<td> 
</td>
</tr>
<tr>
<td> 
Détecteur de flamme KY-026</td>
<td> 
pushbutton</td>
<td> 

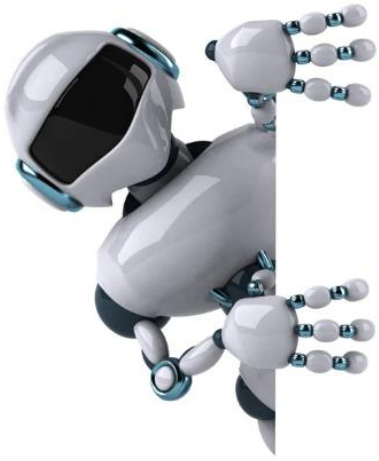
LED RGB</td>
<td> 
Afficheur</td>
</tr>
<tr>
<td> 
plaque d'essai</td>
<td> 
Servomoteur</td>
<td> 
Capteur ultrason</td>
<td> 
Cable</td>
</tr>
<tr>
<td> 
Un potentiomètre 10k</td>
<td> 
Une photorésistance</td>
<td> 
Un écran lcd 16x2</td>
<td> 
Un câble (usb universel)</td>
</tr>
</table>

4.3- Programmation de la carte :

Pour programmer la carte on va utiliser le langage ……

Remarque

Il est possible d'utiliser le simulateur en ligne : **https://wokwi.com/**
Pour programmer la carte ESP32



Partie 2 :
Pilotage d'un objet connecté

Projet 1 : (LED clignotante)

Description :
Faire clignoter une LED (avec un intervalle de 1 seconde) branché sur la carte ESP32 sur le port 23.

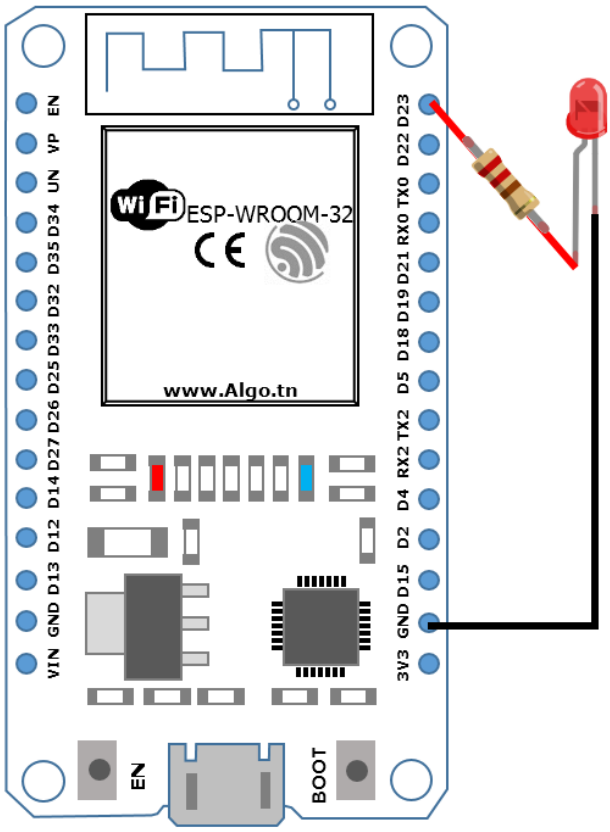
Composants matériels :
Carte ESP32, LED, résistance.

Diode LED


Résistance

Carte Esp32

Branchement :



Code Micro-Python :

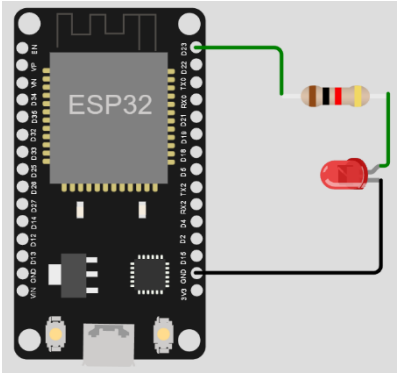
```
import ……
from …… import ……

led = Pin(…… , Pin.OUT)

while True :
        led.value(1)     # Allumer la LED
        time.sleep(1)    # attendre 1 seconde
        led.value(….)    # Eteindre la LED
        time.sleep(…)    # attendre 1 seconde
```

Remarque

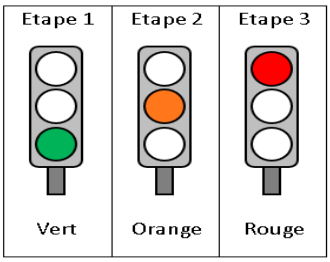
Pour allumer / éteindre la lampe LED, on peut utiliser led.on() et led.off()



Projet 2 : (Feu de Circulation)

Description :
On désire réaliser un feu de circulation en utilisant 3 LEDs (Rouge, Vert, Jaune) qui fonctionne comme suit :

1- Le feu vert (pin4) s'allume 3 secondes puis s'éteint
2- Le feu jaune (pin16) s'allume 1 seconde puis s'éteint
3- Le feu rouge (pin17) s'allume 3 secondes puis s'éteint
4- le programme repart au début et recommence



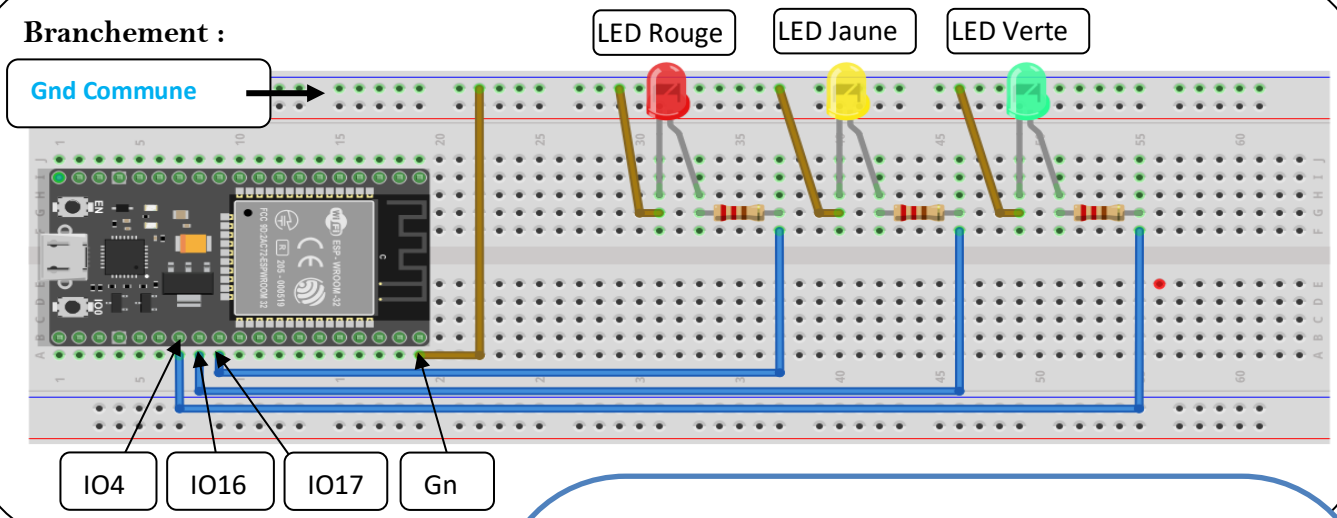
Composants matériels :
1 Carte ESP32
3 LEDs (1 Rouge + 1 Vert + 1 Jaune)
3 résistances
Des Câbles

Diode LED


Résistance

Carte Esp32

Branchement :




```
# Feux de circulation routière
from machine import ……..
from time import ……..
r = Pin(……., Pin.OUT)      # Configuration de la LED rouge
j = Pin(……., Pin.OUT)      # Configuration de la LED jaune
v = Pin(……., Pin.OUT)      # Configuration de la LED verte
while …… :
  j.off()                  # Eteindre la led jaune
  r.on()                   # Allumer la led rouge
  v.off()                  # Eteindre la led verte
  sleep(……..)              # pause de 3 s
  j.off()
  ……                       # Eteindre la led rouge
  ……..                     # Allumer la led verte
  sleep(3)                 # pause de 3 s
  j.on()
  r.off()
  v.off()                  # Eteindre la led verte
  ……..                     # pause de 1 s puis reboucler
```