Anis ELBAHI

Version 1.0

2ème * Sciences

INFORMATIQUE

Anis ELBAHI

Python Programming

Cours, Exercices et Projets

Nom et Prénom : Classe :

Lycée OTHMAN CHATTI M'SAKEN

2022 - 2023

Présentation du manuel :

Ce manuel est destiné aux élèves du niveau **2ème année secondaire section "sciences"** il est reparti sur deux grands axes :

- **Pensée Computationnelle et Programmation**
- **Systèmes et technologies internet (Robotique)**

La première partie contient les 6 chapitres suivants :

Chapitre 1 : Les étapes de résolution d'un problème
Chapitre 2 : Les structures de données
Chapitre 3 : Les structures simples
Chapitre 4 : Les structures de contrôles conditionnelles
Chapitre 5 : Les structures de contrôles itératives (Pour)
Chapitre 6 : Le tableau (Vecteur)

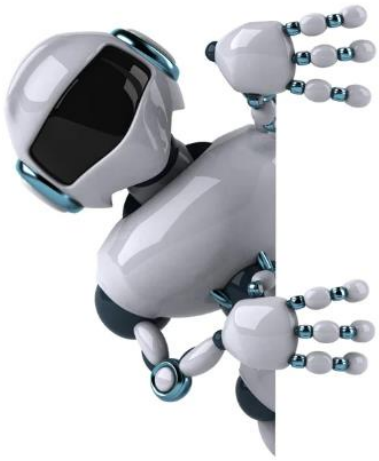
La deuxième partie contient les 2 chapitres suivants :

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 35 exercices et 2 projets robotique pour bien consolider les acquis des élèves durant l'année scolaire.

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

Pensée Computationnelle et Programmation
« Algorithmique et programmation python »



Partie 1 :
Les étapes de résolution d'un problème

1. Les étapes de résolution d'un problème :

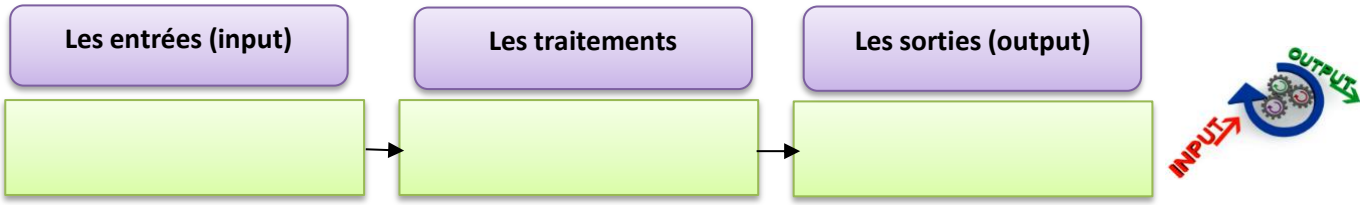
1.1- Présentation :

En informatique, notre objectif est de …… de n'importe quel domaine (mathématique, physique, économie, astronomie, énergie, médecine, enseignement, …)
Un problème peut être de petite taille (calculer la somme de deux entiers) ou de grande taille (contrôler une navette spatiale ou un réacteur nucléaire à distance).

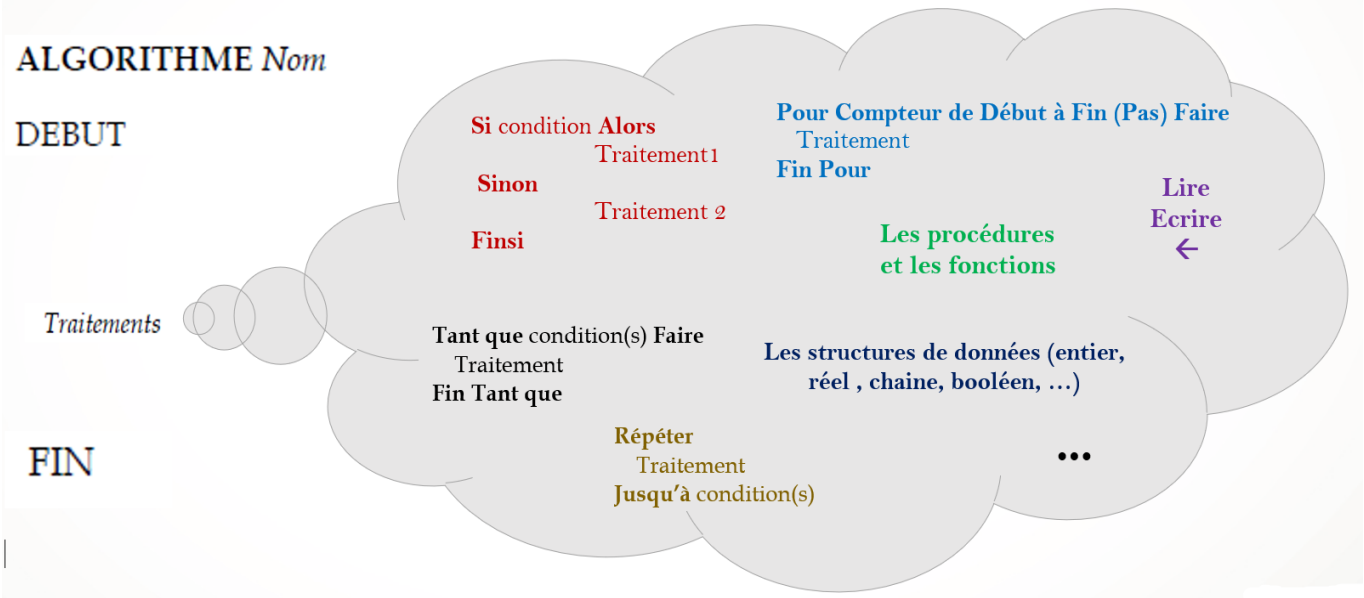
1.2- Les étapes de résolution d'un problème :

pour résoudre un problème informatique, il faut suivre une démarche de 4 étapes.

Dans cette étape il faut dégager les …… , les …… et les …… (les traitements) à effectuer sur les entrées pour avoir les sorties.



Un **algorithme** est …… …… permettant de résoudre un problème. Pour écrire un algorithme il faut respecter la syntaxe



Déclaration des objets

Objet	Type / Nature

Entier
Réel
Booléen
Caractère
Chaine
Tableau

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.

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

Dans cette étape , il faut exécuter et tester le programme

Exercice 1 : (Rectangle)

On désire faire le programme qui demande à l'utilisateur **la longueur (x)** et **la largeur (y)** d'un rectangle et calcule son périmètre (p) et sa surface (s) en utilisant les formules suivantes:

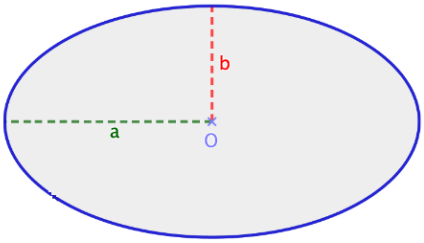
- **Périmètre = (longueur + largeur) *2**
- **Surface = longueur * largeur**

1- Compléter le schéma suivant permettant d'analyser le problème posé.

2- Faire l'algorithme permettant de résoudre le problème posé.
3- Utiliser le langage de programmation Python pour traduire l'algorithme obtenu précédemment.

Exercice 2 : (Ellipse)

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



Exercice 3 : (Moyenne en informatique)

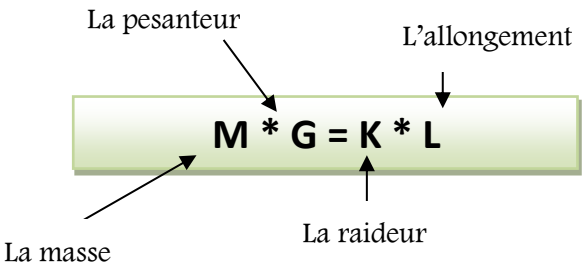
Donner l'analyse, l'algorithme et le programme python qui permet de :

- Lire (saisir) la note de contrôle (NC)
- Lire la note de synthèse (NS)
- Calculer et afficher la moyenne (Moy) obtenue en informatique en appliquant la formule suivante :

Moy = (NC+NS*2)/3

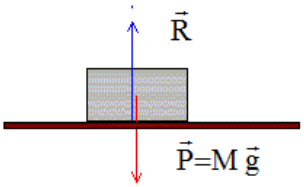
Exercice 4 : (Ressort)

Faire l'analyse, l'algorithme et le programme python permettant de de saisir la raideur **K** du ressort et la masse **M** accrochée au ressort pour calculer et afficher l'allongement **L** correspondant (G est une constante =9.8).



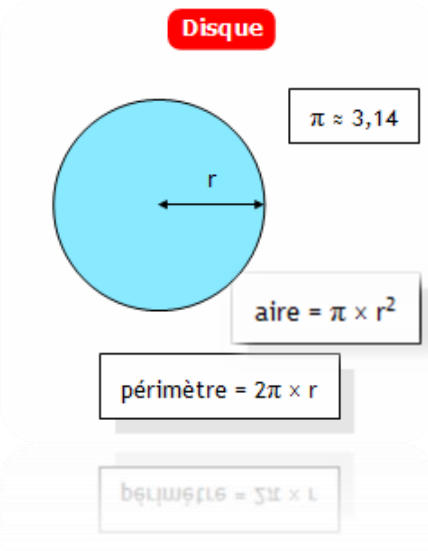
Exercice 5 : (Poids)

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**



Exercice 6 : (Cercle)

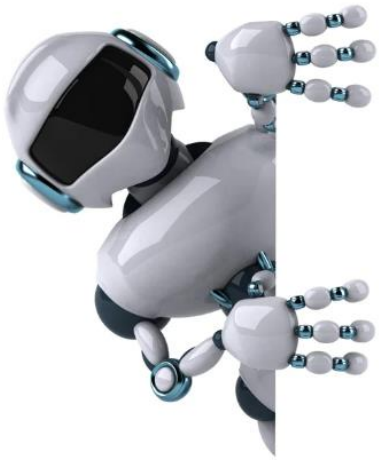
1- Écrire un programme Python qui, à partir de la saisie d'un rayon, calcule et affiche **le périmètre** et **l'aire** d'un cercle.



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.

Pensée Computationnelle et Programmation
« Algorithmique et programmation python »



Partie 2 :
Les structures de données

2. Les structures de données :

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

En programmation, pour écrire un algorithme il faut utiliser des **variables**.

Chaque variable possède :

- un ……
- un ……
- un ……

Remarque

En algorithmique, on peut utiliser d'autre type d'objets qu'on appelle « …… ». 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 ou sans 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 7 : (Opérateurs)

Calculer :

5/2 =	10+4 =	13*5=	10-12=
5 div 2 =	10 div 2 =	5 div 7=	1+(6/2) * 4 =
5 mod 2 =	10 mod 2 =	5 mod 7 =	12 ≥ 10 =

Exercice 8 : (Evaluation d'une expression)

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	
-		4.5 - 2.5	
/		12 / 4	
mod		11 mod 4	
div		11 div 5	
≠		12 ≠ (10+2)	
≤		4 ≤ 4	
∈		5 ∈ [12,6,-3,5,7]	
=		1+2 = 3-0	

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

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

3- Donner la trace d'exécution ainsi que les valeurs finales de X, Y et Z après l'exécution des instructions suivantes :

	Instruction algorithmique	Trace d'exécution		
1	X← 10			
2	Y←2			
3	X← X + Y * 2			
4	Z←X div 2 + Y			
5	Y← (Z + X) % 2			
6	Z ← X *3 + 4 % 2			

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

le réel x = 12 . 75 ……

……

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 9 : (Fonction aléa)

Compléter le tableau suivant par l'instruction python ainsi qu'une valeur possible de x.

	Instruction algorithmique	Instruction en Python	Valeur de X
1	X ← Aléa(10,15)		
2	X ← Aléa(2,10)		
3	X ←Aléa (-2,10)		
4	X ← - Aléa(15,16)		

Exercice 10 : (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
1	x← 15 + 3 * 2 + Ent(5.56)		
2	x← 15. + 3 * 2 + Ent(5.56)		
3	x←(7+ Arrondi (14.36)) div 2		
4	x← 20 >10 *1.5		
5	x← Aléa(10, 20)		
6	x← Aléa(10, 20) > 30		
7	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**

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

Exercice 11 : (Evaluation d'une expression logique)

Pour chaque instruction, donner la valeur de K :

Instruction algorithmique	Valeur de K
K← non ((15+2) = Ent(15.2))	
K← (10>12) et (10 = abs(-10))	
K← 1*2*3*0 = 0	
K← 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 numéro qu'on appelle code ASCII.

Remarque

Un caractère peut être :

- Alphabétique (Majuscule ou minuscule)
- Chiffres : "0","1","2","3","4", ... ,"9"
- Symbole : " ","*","/","=","+",";", ... ,")" ,"%"

Remarque

Exemple 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 .....
- L'indice du premier caractère d'une chaine est .....

Remarque

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

Exercice 12 : (Concaténation des chaines)

En utilisant les chaines Ch1, Ch2 et Ch3 donner pour chaque cas la chaine résultante :

Ch1="Python" **Ch2=**"Algo*123" **Ch3=**"2023"	
Ch1[0]= "P"	Ch1[3]=
Ch2[4]=	Ch1[0]+Ch2[0]+Ch3[2]=
Ch1+Ch3=	Ch2+"info"+Ch1[0]+Ch1[1] =

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"

Exercice 13 : (Fonctions de manipulation d'une chaine - algo)

En utilisant les chaines Ch1, Ch2 et Ch3 donner pour chaque instruction la valeur de X :

	Ch1="Algorithmique" **Ch2=**"Python" **Ch3=**"4502"	
	Instruction algorithmique	**Valeur de X**
1	X← long(Ch2)	
2	X← long(Ch2+Ch3)	
3	X← pos("Go",Ch1)	
4	X← pos("i",Ch1)	
5	X← estnum(Ch3)	
6	X← valeur(Ch3) + 4	
7	X← convch(246)	
8	X←sous_chaine(Ch2, 2, long(Ch2))	
9	X←efface(Ch2, 2, long(Ch2))	
10	X← majus("Anis")	

Exercice 14 : (Fonctions de manipulation d'une chaine - Py)

Soient les chaines suivantes, donner l'équivalent en python de chacune des instructions suivantes :

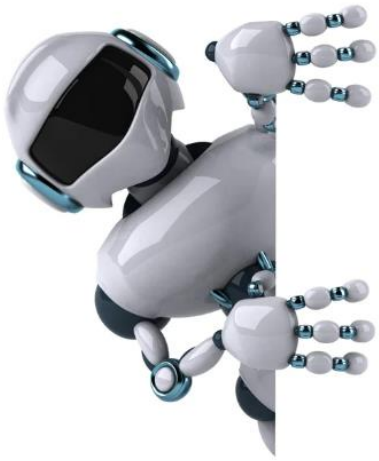
	Ch1="Algorithmique" **Ch2=**"Python" **Ch3=**"4502"	
	Instruction algorithmique	**Python**
1	X← long(Ch2)	
2	X← long(Ch2+Ch3)	
3	X← pos("Go",Ch1)	
4	X← pos("i",Ch1)	
5	X← estnum(Ch3)	
6	X← valeur(Ch3) + 4	
7	X← convch(246)	
8	X←sous_chaine(Ch2, 2, long(Ch2))	
9	X←efface(Ch2, 2, long(Ch2))	
10	X← majus("Anis")	

Exercice 15 : (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()
```

Pensée Computationnelle et Programmation
« Algorithmique et programmation python »



Partie 3 :
Les structures simples

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 16 : (Affichage)

Pour chaque instruction, donner le résultat affiché sachant que ch = "Peace" et x = 65 :

	INSTRUCTION ALGORITHMIQUE	RESULTAT AFFICHE
1	**Ecrire ("Bonjour")**	
2	**Ecrire (x)**	
3	**Ecrire ("le code de A =", x)**	
4	**Ecrire_nl (ch) , Ecrire (x) , Ecrire ("Ok")**	

Exercice 17 : (Moyenne en informatique)

On désire calculer la moyenne obtenue par un élève en informatique pour cela on vous demande de faire l'algorithme puis le programme python permettant de saisir la note de contrôle et celle de synthèse pour calculer et afficher la moyenne obtenue.

Exercice 18 : (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 ☐

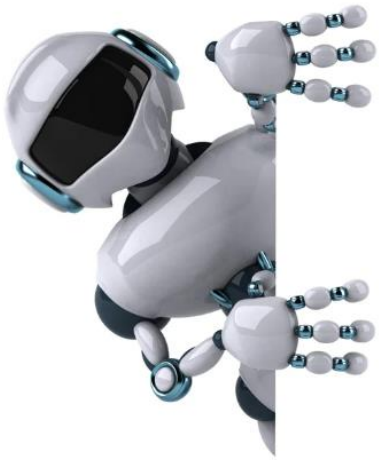
2. **L'instruction R← arrondi (12.75) permet d'effectuer à la variable R**
   - L'entier 12 ☐
   - L'entier 13 ☐
   - L'entier 14 ☐

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("123") affecte à T:**
   - 123 ☐
   - 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 ☐

Pensée Computationnelle et Programmation
« Algorithmique et programmation python »



Partie 4 :
Les structures conditionnelles

4. Les structures conditionnelles :

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

Exercice 19 : (Passable ou redoublant)

On vous demande de faire un algorithme nommé "**scolaire**" qui permet de :

- Saisir la moyenne de chaque trimestre (T1, T2 et T3)
- Calculer la moyenne générale obtenue (Moy) en appliquant la formule suivante :

Moy = (T1+T2*2+T3*2) / 5

- Afficher à l'écran si l'élève est admis ou refusé sachant qu'un élève est considéré admis si sa moyenne est ≥ 10.

Exercice 20 : (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- Traduire votre algorithme en Python

Exercice 21 : (Vérifier le signe)

1- On vous demande de faire un algorithme nommé "**signe**" qui :
   - Permet de saisir un entier x
   - Vérifier et afficher s'il est négatif, nul ou positif .
2- Traduire votre algorithme en Python

Les structures conditionnelles permettent de faire un choix parmi plusieurs (et de rendre un programme informatique plus intelligent) . En programmation, on peut distinguer **3 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`
		`  ...`

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 22 : (Nombre d'Armstrong)

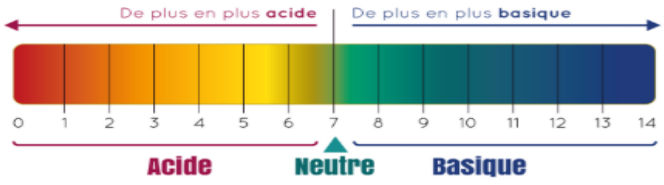
On appelle nombre d'Armstrong, un nombre positif de 3 chiffres qui a la propriété d'être égal à la somme des cubes de ses chiffres comme N =153 car 153 = $(1)^3 + (5)^3 + (3)^3$.

Travail demandé :

Ecrire l'algorithme d'un programme nommé « **Armstrong** » qui permet de saisir un entier N et vérifier et afficher si l'entier N est d'Armstrong ou non.

Exercice 23 : (Potentiel Hydrogène)

Faire l'algorithme nommé « **nature_liquide** » puis le programme Python qui permet de saisir la valeur de PH (Potentiel Hydrogène) d'un liquide pour afficher sa nature (base, neutre ou acide) en se basant sur l'échelle suivant.



Échelle de pH.

Exercice 24 : (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

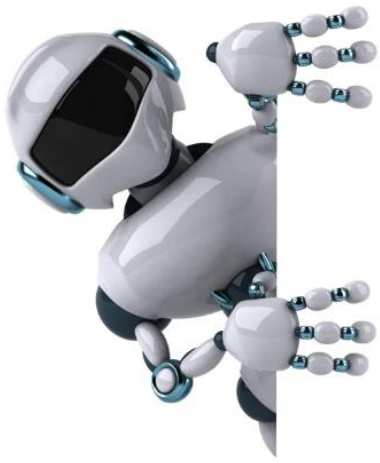
```
A ←2
A ←0
Si   A=2  alors
     B← 3
Sinon si A=0 alors
     B← 2
Sinon
     B← 1
Fin si
```

A	B

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

A	B

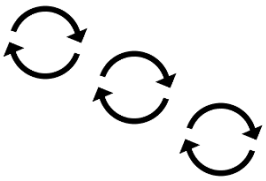
Pensée Computationnelle et Programmation
« Algorithmique et programmation python »



Partie 5 :
Les structures de contrôles itératives (La boucle Pour)

5- Les structures itératives

Les structures itératives ou structures répétitives ou ..... permettent de ..... un traitement un certain nombre de fois.



La boucle « Pour »:

En Algorithmique	En Python
i : compteur, *Vi : Valeur initiale du compteur* **Pour i de Vi à Vf (pas) Faire** Traitement **Fin Pour** *Vf : Valeur finale du compteur*	**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*

Exercice 25 : (« Puissance » de la boucle Pour)

1- Donner l'algorithme permettant d'afficher le mot "Bonjour" 20 fois à l'écran.
2- Modifier l'algorithme précédant pour qu'il affiche le même mot "Bonjour" 2000 fois.

Exercice 26 : (Somme des 100 premiers entiers)

Donner l'algorithme puis le script python permettant de calculer et d'afficher la somme des 100 premiers entiers (1→100)

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.

2- Le nombre d'itérations de la boucle « Pour » est : **|Vf-Vi| + 1**
3- Le **« Pas »** peut être Positif ou négatif, par défaut, le **« Pas »** est égal à 1.
4- Le parcours peut être croissant (Début > Fin) ou décroissant (Fin<Début) dans les deux

Exercice 27 : (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			

Exercice 28 : (Nombre de voyelles dans une chaine)

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

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

Exercice 29 : (Somme des chiffres dans une chaine)

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

- Saisir une chaine **ch**
- 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 30 : (Deviner le résultat)

Cochez la bonne réponse

x=0 **for i in range(1,4):** **x=x+i** **print('x=',x)**	Le programme affiche : ☐ x= 6 ☐ x= 15 ☐ x= 10
y=1 **for i in range(1,4):** **y=y*i** **print('y=',y)**	Le programme affiche : ☐ y=20 ☐ y=25 ☐ 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

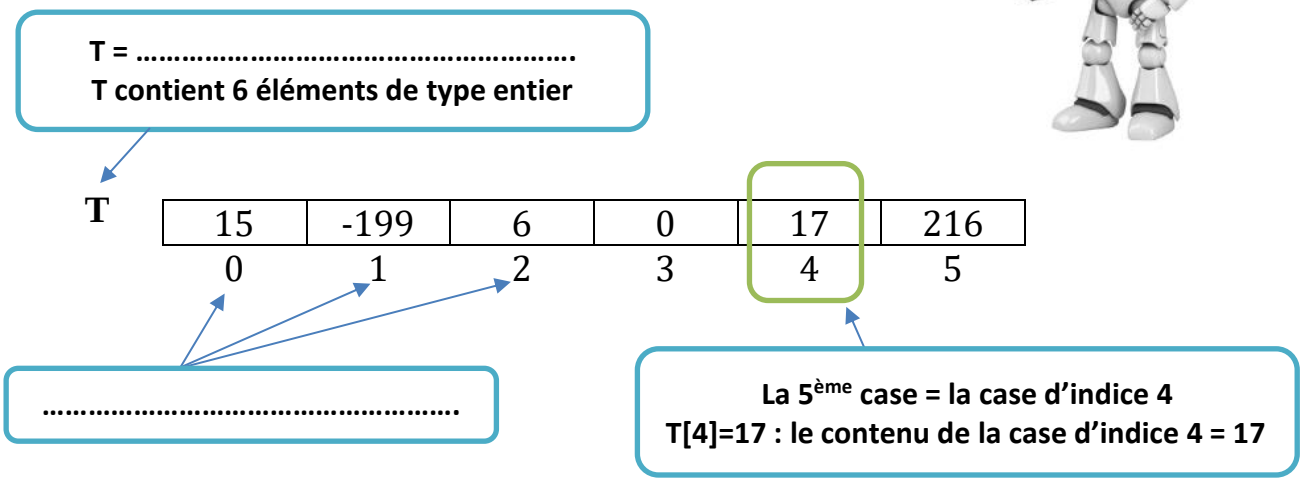
Pensée Computationnelle et Programmation
« Algorithmique et programmation python »

Partie 6 :
Le tableau (Vecteur)

6- Le tableau :

6.1 - Présentation :

Appelé aussi tableau à une dimension (ou vecteur). Il permet le regroupent d'un ensemble d'éléments de même type.



6.2- 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

6.3 - Implémentation en Python :

Pour implémenter un tableau en Python, nous utiliserons la fonction array de la bibliothèque numpy 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 31 : (Déclaration d'un tableau)

Soit le vecteur T suivant remplit par 5 réels comme suit :

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

1- Déclarer le tableau T en algorithmique
2- Déclarer le tableau T en Python
3- Donner le code python permettant d'afficher le contenu du tableau (utiliser une boucle pour parcourir le tableau).

Exercice 32 : (Manipulation simple d'un tableau)

1- Donner l'algorithme d'un programme nommé « **liste**» qui permet de faire les tâches suivantes :
   - Remplir un tableau T par 10 entiers
   - Afficher le contenu du tableau T
   - Chercher et afficher le maximum du tableau T
2- Donner le script python de votre programme

Exercice 33 : (Manipulation simple d'un tableau)

Donner l'algorithme d'un programme qui permet de faire les tâches suivantes :

- Remplir un tableau T par 15 entiers de façon aléatoire sachant que $2 \leq T[i] \leq 20$
- Calculer et afficher la somme des éléments qui se trouvent dans le tableau

Exercice 34 : (Recherche dans un tableau)

On désire faire un programme (algorithme et python) qui permet de :

1- Remplir un tableau T par N entiers (supposons que $2 \leq N \leq 20$)
2- Introduire un entier X quelconque
3- Vérifier et afficher si l'entier X existe ou non dans le tableau T

Exemple:

Pour le tableau T suivant de taille N =5:

15	10	3	5	172
0	1	2	3	4

Pour x = 5 , le programme affiche le message : **5 existe dans le tableau T**
Pour x = 7 , le programme affiche le message : **7 n'existe pas dans le tableau T**

Exercice 35 : (Manipulation avancée d'un tableau)

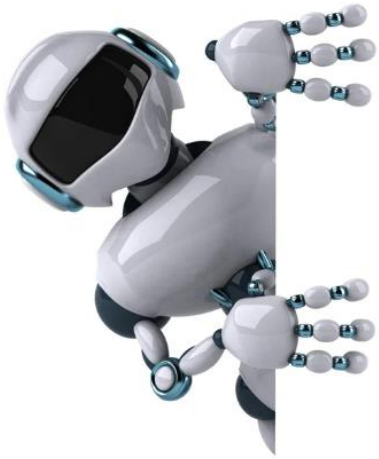
Donner l'algorithme d'un programme nommé « **Armstrong** » qui permet de :

1- Saisir la taille N du tableau T : **4 ≤ N < 15**
2- Remplir un tableau T par N entiers (supposons 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 : $153 = 1^3+5^3+3^3$

Systèmes et technologies internet (Robotique)



Partie 1 :

Présentation de la robotique

1- La robotique :

1.1 - Présentation :

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

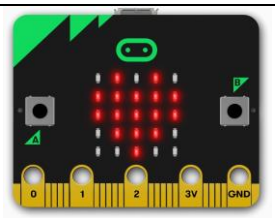
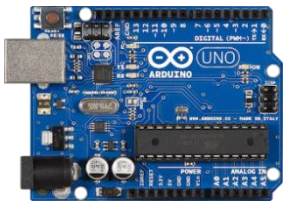
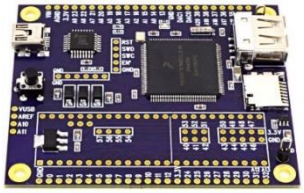
1.2 - Domaines d'application de la robotique :

Les robots sont à peu près partout.

- ……
- ……
- ……
- ……
- ……
- ……

1.3 - Enseignement de la robotique :

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

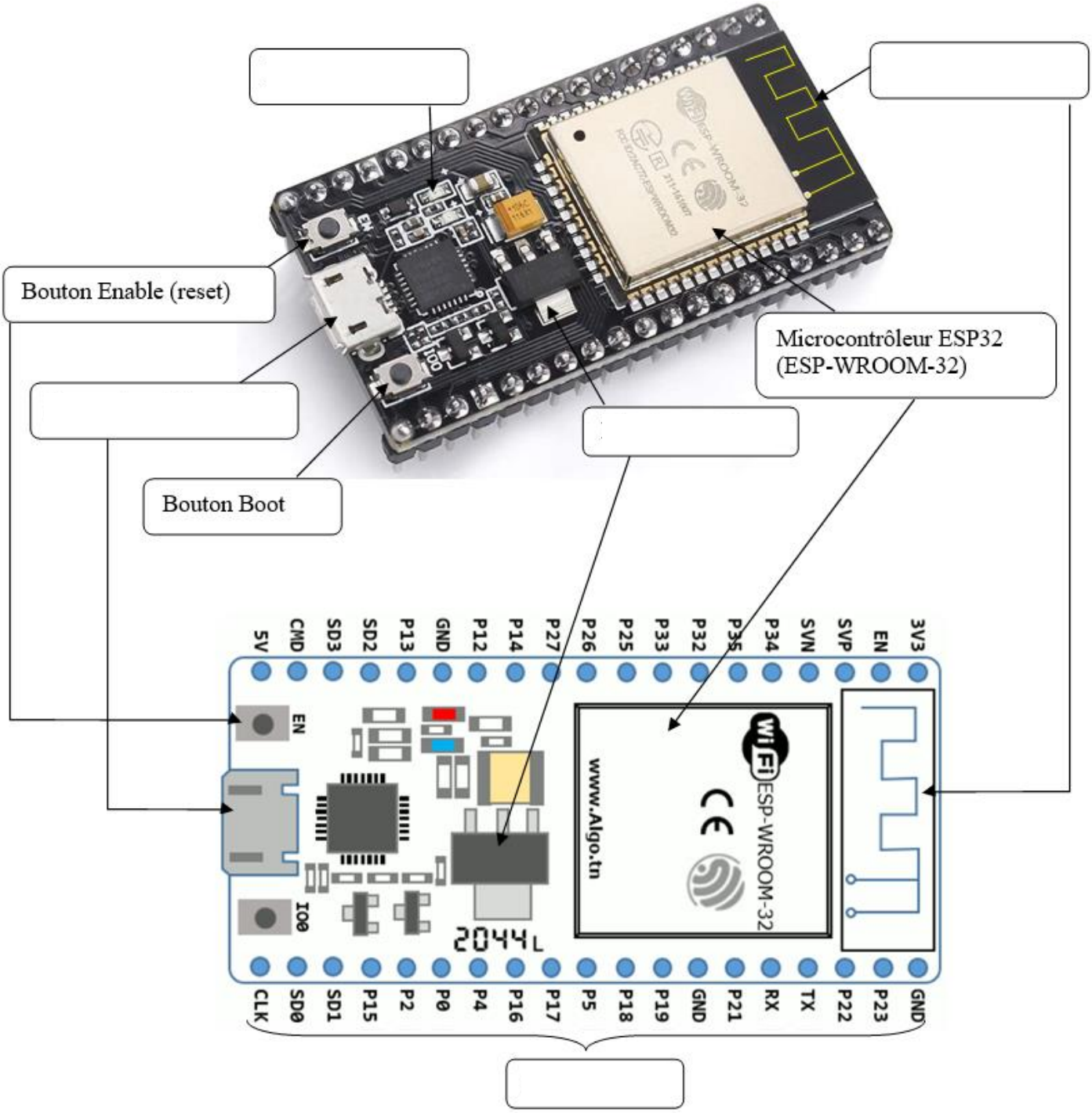
			
La carte Micr- bit	**La carte Arduino**	**La carte Raspberry**	**La carte Lattepanda**
			
La carte Pycom	**La carte teensy**	**La carte Mcore**	

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

1.4 - La carte ESP32 :

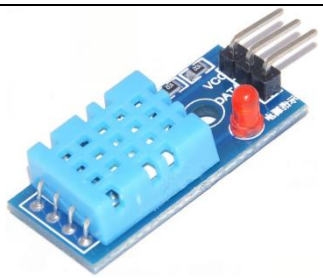
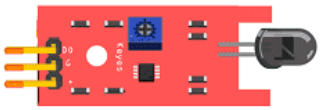
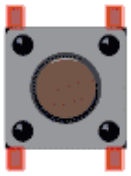
a) Présentation de la carte :

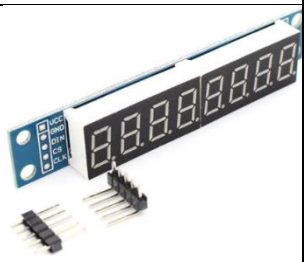
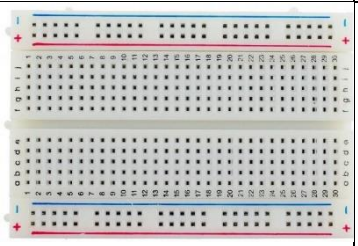
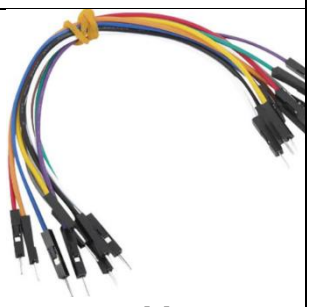
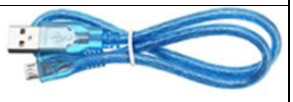
La carte ESP32 est une petite …… développée par la société Espressif Systems.



b) 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> 

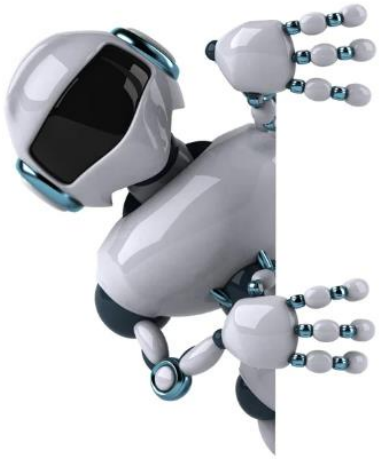
LED RGB</td>
<td> 
Afficheur</td>
</tr>
<tr>
<td> 
plaque d'essai</td>
<td> 
Servomoteur</td>
<td> 
Capteur ultrason</td>
<td> 
Cable</td>
</tr>
<tr>
<td> 
Un potentiomètre 10k</td>
<td> 
Une photorésistance</td>
<td> 
Un écran lcd 16x2</td>
<td> 
Un câble (usb universel)</td>
</tr>
</table>

c) 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

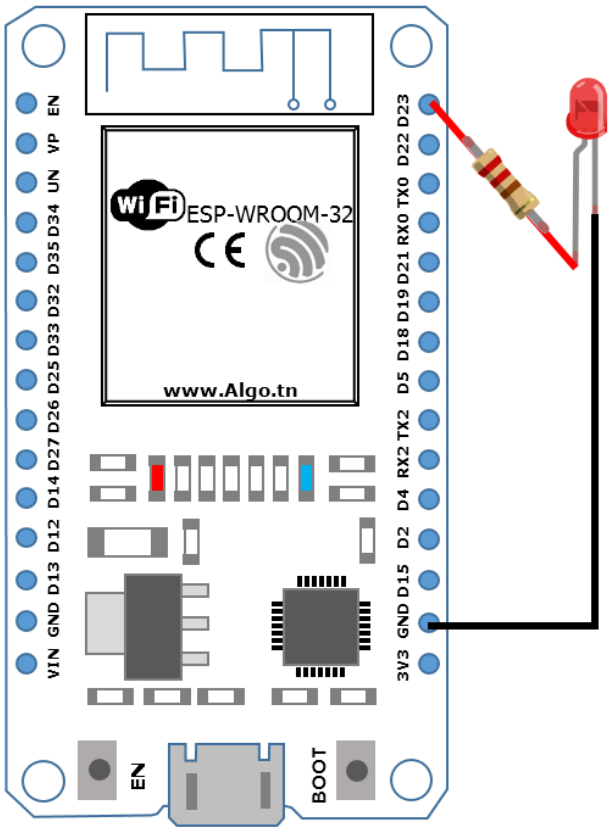
Projet 1 : (LED clignotante)

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.

Carte Esp32

Branchement :



Code Micro-Python :

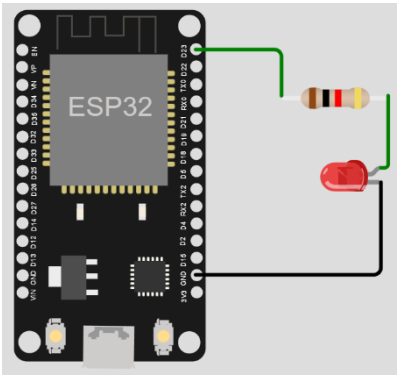
```
import ……
from …… import ……

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

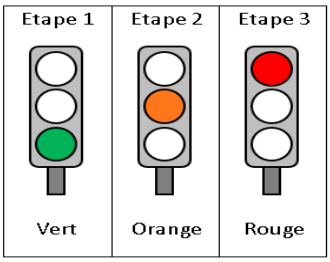
Pour allumer / éteindre la lampe LED, on peut utiliser led.on() et led.off()



Projet 2 : (Feu de Circulation)

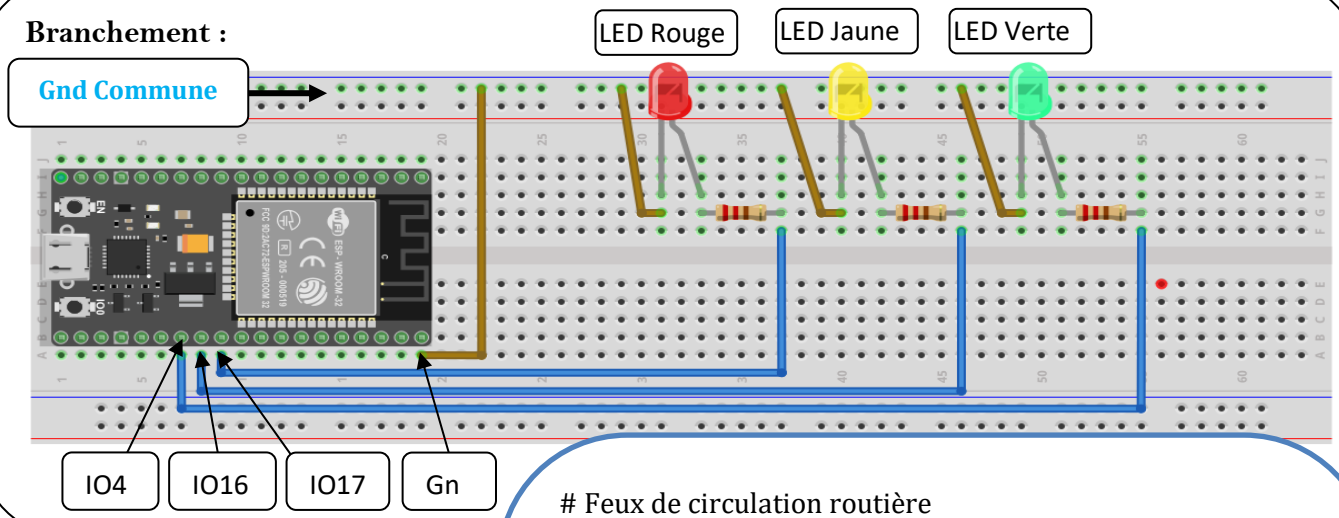
Description :
On désire réaliser un feu de circulation en utilisant 3 LEDs (Rouge, Vert, Jaune) qui fonctionne comme suit :

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

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
```