REPUBLIQUE TUNISIENNE
MINISTERE DE L'EDUCATION
DIRECTION RÉGIONALE DE L'ENSEIGNEMENT NABEUL

Cours
Algorithme et Programmation
2ème TI

Auteur

Riadh Ben Daoud
Enseignant informatique
Lycée Elmida

Sommaire

- **Partie 1 : Démarche de résolution de problèmes 1**
  - **I.** Introduction
  - **II.** Les étapes de résolution d'un problème

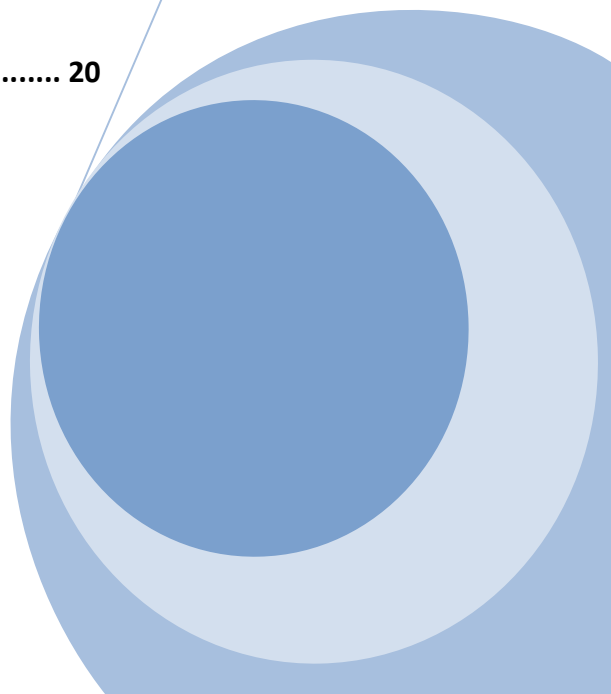
- **Partie 2 : Les structures simples 3**
  - **I.** L'opération d'entrée
  - **II.** L'opération de sortie
  - **III.** L'opération d'affectation

- **Partie 3 : Les structures de données 6**
  - **I.** Les constantes
  - **II.** Les variables
  - **III.** Les types de données
  - **IV.** Le tableau à une dimension (Vecteur)

- **Partie 4 : Les structures de contrôle conditionnelles 12**
  - **I.** La structure conditionnelle simple
  - **II.** La structure conditionnelle généralisée
  - **III.** La structure conditionnelle à choix

- **Partie 5 : Les structures de contrôle itératives 16**
  - **I.** La structure itérative complète
  - **II.** La structure itérative à condition d'arrêt

- **Partie 6 : Les sous programmes 20**
  - **I.** Les fonctions
  - **II.** Les procédures



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

1. 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**.

a. Spécifier les différentes données nécessaires pour résoudre ce problème

b. Proposer le traitement nécessaire pour avoir le résultat voulu à partir les données

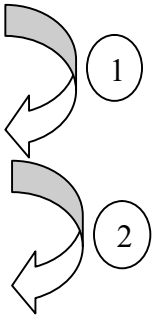
c. Indiquer le résultat à obtenir.

d. Ecrire un algorithme permettant de résoudre ce problème.

e. Implémenter cet algorithme en Python.

➢ **Solution :**

- **Données :**
  .....
  .....
- **Traitement :**
  .....
  .....
- **Résultat :**
  .....
  .....



Algorithme	Implémentation en Python
Algorithme Rectangle **Début** .... **Fin**	....

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

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

Résultat d'exécution

```
Taper Lo :2.5
Taper La :1.25
La surface =  3.125  Le périmètre =  7.5
```

2. Activité 2:

On se propose de calculer et d'afficher la moyenne annuelle MA d'un élève ayant les notes suivants NT1, NT2 et NT3.

a. Spécifier les différentes données nécessaires pour résoudre ce problème

b. Proposer le traitement nécessaire pour avoir le résultat voulu à partir les données

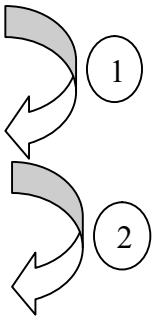
c. Indiquer le résultat à obtenir.

d. Ecrire un algorithme permettant de résoudre ce problème.

e. Implémenter cet algorithme en Python.

➢ **Solution :**

- **Données :**
  ……
  ……
- **Traitement :**
  ……
  ……
- **Résultat :**
  ……
  ……



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

▪ **T.D.O :**

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

Résultat d'exécution

```
Donner NT1:15.5
Donner NT2:13.28
Donner NT3:14.18
La moyenne annuelle =  14.084
```

Partie 2 : Les structures 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. Exemple :

En Algorithme	En Python
Lire (n)	n = **int** (**input**( ))

3. 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
.....	.....
.....	.....
.....	.....

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 :

Algorithme	Python
Variable ← valeur	Variable = valeur

3. Exemple :

Algorithme	Python	Commentaire	Résultat
x←5	.....	x reçoit 5	x = ...
a←x+3	.....	a reçoit la valeur de l'expression x+3	a = ...
x←x-2	.....	x reçoit la valeur de l'expression x-2	x = ...

4. Remarque :

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

5. 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	6
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 ?

.....

6. 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 ?

.....

7. 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 :**

Algorithme	Implémentation en Python
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....

▪ **T.D.O :**

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

Partie 3 : Les structures de données

I. Les constantes et les variables :

1. 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
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....

▪ **T.D.O :**

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

2. Activité 2 :

Pour chaque rayon donné, calculez la surface et le périmètre du cercle correspondant on exécutant le programme de l'activité1

PI	R	S	P
3.14	2.5	.....	.....
3.14	1.25	.....	.....
3.14	4.5	.....	.....

☞ On constate que PI est une **constante** tandis que R, S et P prennent des valeurs différentes. R, S et P sont appelés des **variables**.

II. Les types de données :

1. Le type Entier (int) :

a. Définition :

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

b. 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 .....
$n \in [5,20]$	5<=n<=20	Appartient	5<=3<=20 renvoie .....
X^n	X ** n	Puissance	**5 ** 2** = ...
MOD	%	Modulo	**5 % 2** = ...
DIV	//	Division entière	**5 // 2 = ...**

c. 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
.... ▪ **T.D.O :** Objet | Type/Nature ..... | .... | .... |	....

2. Le type Réel (float) :

a. Définition :

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

b. Les opérations arithmétiques et relationnelles sur les réels :

Les mêmes opérations que les entiers sauf DIV et MOD.

c. Les fonctions prédéfinies sur les réels :

Fonctions (en algorithme)	Fonctions (en Python)	Exemple
Ent (x)	**int** (x)	**int** (3.51) = ...
Arrondi (x)	**round** (x)	**round** (3.49) = ... **round** (3.51) = ...
Abs (x)	**abs** (x)	**abs** (-3) = ...
RacineCarré (x)	**sqrt** (x)	**sqrt** (25) = ...
Aléa ( )	**random ( )**	Renvoie un réel aléatoire de [... , ... [
Aléa (a, b)	**uniform (a, b)**	Renvoie un réel aléatoire de [... , ...]
Aléa (m, n)	**randint (m, n)**	Renvoie un entier aléatoire de [... , ...]

d. 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) = $\sqrt{(x2-x1)^2+(y2-y1)^2}$

➢ **Solution :**

Algorithme	Implémentation en Python
.... ▪ **T.D.O :** **Objet** / **Type/Nature** ..... / .... / .... /	....

3. Le type Booléen (bool) :

a. Définition :

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

b. Opérations logiques sur les Booléens :

En Algorithme	En Python	Rôle
NON	NOT	Négation
OU	OR	Disjonction
ET	AND	Conjonction

La table de vérité qui manipule ces opérateurs est :

x	y	NON(x)	x ET y	x OU y
V	V	.....	.....	.....
V	F	.....	.....	.....
F	V	.....	.....	.....
F	F	.....	.....	.....

c. 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):

a. Définition :

Un caractère (chiffre où lettre où symbole) est représenté le caractère lui-même mis entre guillemets

Exemple : "A", "a", "+",…

Remarque :

- Une variable de type caractère contient un caractère est un seul.
- L'espace " " est le caractère blanc.

b. Opération sur les caractères :

Les opérations usuels : +, =, <, >, <=, >=, <>.

Exemple :

"A" < "B" est une proposition vrai.

"a" >"b" est une proposition fausse.

Car tous les caractères sont ordonnés selon leur code ASCII.

c. Fonctions prédéfinis sur les caractères :

Fonction (en algorithme)	En Python	Exemples
ORD(c)	ORD (c)	ORD ("A") = 65 ORD ("E") = ... ORD ("a") = 97 ORD ("d") = ...
CHR (n)	CHR (n)	CHR (65) = ... CHR (66) = ... CHR (97) = ... CHR (ORD ("E")) = ...

Remarque :

- "x" : désigne le caractère "x".
- x : désigne l'objet x constant ou variable.

5. Le type chaîne de caractère (str):

a. Définition :

C'est une succession de n caractère (lettre, symbole, chiffre) avec n ≥ 0.

Remarque :

- Si n = 0 alors la chaîne est dite vide ("" : chaîne vide).
- Les valeurs chaîne de caractères sont définies entre guillemets

b. Manipulation de chaîne de caractère :

On peut accéder en lecture et en écriture au $i^{ème}$ caractère d'une chaîne Ch en utilisant la notation **CH[i]** avec 1 ≤ i ≤ Long (Ch).

Exemple : Soit Ch ← "Bonjour papa"

- Ch [0] = "B"
- Ch [1] = "o"
- Ch [:3] = "Bon" renvoie les trois premiers caractères
- Ch [3:] = "jour papa" renvoie Ch on élimine les trois premiers caractères
- Ch [::3] = "Bjra"
- Ch [3:7] = "jour" renvoie une sous chaine de Ch de la position 3 jusqu'à la position 7-1=6

c. Fonctions prédéfinis sur les chaînes de caractère:

Fonctions en algorithme	Fonctions en Python	Description
Long (ch)	**len** (ch)	Retourne le nombre de caractères de la chaine ch.
Pos (ch1, ch2)	ch2**.find** (ch1)	Retourne la première position de **ch1** dans **ch2**. Si **ch1** n'existe pas dans **ch2**, retourne -1
Convch(x)	**str**(x)	Convertit un nombre **x** en chaine.
Estnum(ch)	ch.**isnumeric()**	Retourne Vrai si la chaine ch est convertible en numérique, Faux dans le cas contraire.
Valeur(ch)	**int**(ch)	Retourne la conversion d'une chaine ch en numérique si c'est possible.
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** exclue.
Effacer(ch, d, f)	ch[:d]+ch[f:]	Efface des caractèresde la chaine ch à partir de la position **d** jusqu'à la position **f** exclue.
Majus(ch)	ch**.upper()**	Convertit la chaine **ch** en majuscule.

- **N.B. :** On utilise l'opérateur **+** pour concaténer deux chaînes.

III. Le tableau à une dimension (Vecteur) :

1. Définition :

Un vecteur est une structure de données permettant de ranger un nombre fini d'éléments de même type.

2. Déclaration d'une variable de type vecteur :

<table>
<tr><th>En algorithme</th><th>Implémentation en Python</th></tr>
<tr><td>▪ TDO
<table><tr><th>O</th><th>T/N</th></tr><tr><td>T</td><td>tableau de 20 entiers</td></tr></table></td><td>from numpy import array
T =array ([int ( )] * 20)</td></tr>
</table>

➢ **Remarque :**

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

3. Accès aux éléments d'un vecteur :

L'accès à chaque élément ce fait par le biais d'un **indice**.

➢ **Exemple :** 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] = 10.5
- T [1] = 0.25
- T [9] = 12.00

4. Modifier les éléments d'un vecteur :

➢ **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	20	2	10	50	70
	0	1	2	3	4

Partie 4 : Les structures de contrôle conditionnelles

I. La structure de contrôle conditionnelle simple :

1. Activité 1 :

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

⌖ **Exemples :**

- 25 est impaire
- 36 est paire

D'après sur ce qu'on a déjà vu, on ne peut pas résoudre ce type de problème car on a amènera à décider si l'entier n est paire ou impaire, donc on a intérêt de définir une nouvelle structure qui permet de résoudre ce problème.

2. 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).

3. Vocabulaire et syntaxe :

Algorithme	Implémentation en Python
Si condition **alors** <traitement 1> **Si non** <traitement 2> **Fin Si**	**if** condition **:** <traitement 1> **else :** <traitement 2>

4. 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.
- Si traitement 2 est vide, on parle de structure conditionnelle simple réduite qui à la syntaxe suivante :

Algorithme	Implémentation en Python
Si condition **alors** <traitement 1> **Fin Si**	**if** condition **:** <traitement 1>

2. Solution de l'activité 1 :

Algorithme	Implémentation en Python
…… …… …… …… …… …… …… …… …… …… ▪ **T.D.O :** **Objet** / **Type/Nature** …… / …… …… / ……	…… …… …… …… …… …… …… …… …… ……

II. La structure de contrôle généralisée :

1. 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

Remarquez bien qu'on a plus que deux traitements donc on a intérêt de définir une nouvelle forme de structure de contrôle conditionnelle qui nous permet de résoudre ce type de problème.

2. 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.

3. Vocabulaire et syntaxe :

Algorithme	Implémentation en Python
Si condition1 **alors** <traitement1> **Sinon Si** condition2 **alors** <traitement2> ... **Sinon Si** conditionN-1 **alors** <traitement N-1> **Sinon** <traitement N> **Fin Si**	**if** condition1 **:** <traitement 1> **elif** condition2 **:** <traitement 2> ... **elif** condition N-1 **:** <traitement N-1> **else :** <traitement N>

4. Remarque :

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

5. Solution de l'activité 2 :

Algorithme	Implémentation en Python
.... ▪ **T.D.O :**	....

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

6. Application :

Ecrire un algorithme qui permet de saisir le numéro du jour puis afficher son nom (1← Dimanche ; 2 ← Lundi ; … ; 7 ← Samedi)

7. Solution de l'application :

Algorithme	Implémentation en Python
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....

- **T.D.O :**

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

III. Structure de choix :

1. Activité 3 :

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 seul variable, donc on a intérêt de définir une nouvelle structure qui offre une meilleure lisibilité dans ce cas.

2. Définition :

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

3. Vocabulaire et syntaxe :

Algorithme	Implémentation 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 : <traitement 2> … **case_ :** <traitement N>

4. Remarque :

- Cette structure offre une meilleure lisibilité de la solution que la forme généralisée.
- Le sélecteur doit être de type scalaire discret et non réel.
- Chaque bloc peut comporter une ou plusieurs instructions.
- La clause sinon et son traitement sont facultatifs dans la structure.

5. Solution de l'activité 2 avec la structure de choix :

Algorithme	Implémentation en Python
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....

- **T.D.O :**

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

Partie 5 : Les structures de contrôle itératives

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

1. Activité 3 :

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

☞ 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 répété l'exécution d'une instruction un nombre connu de fois.

2. Définition :

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

3. Vocabulaire et syntaxe :

Algorithme	Python
Pour <compteur> **de** <Vi> **à** <Vf> **Faire** <traitement1> **Fin pour**	**for** <compteur> **in range** (Vi, Vf)**:** <traitement1> **else :** <traitement 2>

4. Remarque :

- Le compteur est une variable de type scalaire énuméré (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 chaque répétition 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.
- La clause **else** est facultatif et elle n'est exécutée que lorsque on a attendre la valeur finale Vf-1 de la boucle **for**
- On peut utiliser l'instruction **break** pour sortir de la boucle et passer à l'instruction suivante
- On peut utiliser l'instruction **contenue** pour passer directement à l'itération suivante

5. Solution de l'activité 3 :

Algorithme	Implémentation en Python
…… …… …… …… …… …… ▪ **T.D.O :** **Objet** / **Type/Nature** …… / ……	…… …… …… …… …… ……

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

1. Activité 4 :

Ecrire un programme qui permet de saisir un mot de passe et confirmer à nouveau jusqu'à avoir le mot de passe est confirmé correctement.

☞ Si on utilise les structures de contrôles conditionnelles l'utilisateur peut se tromper dans la saisir de mot de passe, d'où la nécessité d'une nouvelle structure appelé la structure de contrôle itérative à condition d'arrêt que nous permet de répété l'exécution d'une instruction un nombre inconnu de fois.

2. Définition :

La structure Répéter … Jusqu'à 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 au moins une fois.

3. Vocabulaire et syntaxe :

Algorithme	Implémentation en Python
Répéter <traitement> **Jusqu'à** <condition de sortie>	**While Not** <condition de sortie> **:** <traitement>

4. Remarque :

- La condition d'arrêt est considérée comme une **condition de sortie** de la boucle car, une fois elle est vérifier 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.

5. Solution de l'activité 4 :

Algorithme	Implémentation en Python
…… …… …… …… …… …… …… …… …… …… ▪ **T.D.O :** **Objet** / **Type/Nature** …… / …… …… / ……	…… …… …… …… …… …… …… …… …… ……

6. 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] :

1. Activité 5 :

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**.

Algorithme
Algorithme PGCD **Début** Ecrire ("m= "), Lire (m) Ecrire ("n= "), Lire (n) **Répéter** Si m>n Alors M←m-n Sinon N←n-m Fin Si **Jusqu'à** (m=n) Ecrire ("PGCD = ", m) **Fin**

☞ 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 dés que la condition m=n est vérifiée. Donc on a intérêt des définir une nouvelle structure qu'elle peut résoudre ce type de problème.

2. 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.

3. Vocabulaire et syntaxe :

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

4. Remarque :

- La condition d'arrêt est considérée comme une **condition d'entrée** car, tant que elle est vérifier 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.

5. Solution de l'activité 5 :

Algorithme	Implémentation en Python
…… …… …… …… …… …… …… …… …… …… …… …… …… ▪ **T.D.O :** **Objet** / **Type/Nature** …… / …… …… / ……	…… …… …… …… …… …… …… …… …… …… …… …… ……

Partie 6 : Les sous programmes

I. Introduction :

Nous avons vu jusqu'à maintenant les différentes structures nécessaires pour résoudre un tel problème, mais dés 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.

II. Les fonctions :

1. Activité 1 :

Ecrire un algorithme et son implémentation en Python d'un programme qui permet de calculer puis d'afficher le nombre de combinaison de P objets parmi N.

C_N^P = N! / (P!(N-P)!) ; N, P sont deux entiers strictement positifs avec N≥P.

✧ **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.

2. Vocabulaire et syntaxe :

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

Implémentation en Python
def Nom_fonction (pf1, pf2 , … , pfn) **:**   <Traitement> **return** Résultat

3. Solution de l'activité 1 :

Algorithme	Implémentation en Python
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
▪ **T.D.O :**	
Objet | **Type/Nature**	
..... |	
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
▪ **T.D.O :**	
Objet | **Type/Nature**	
..... |	

4. Caractéristiques

- Une fonction est un sous programme qui retourne une valeur d'un type identique à celui de la fonction.
- Une fonction comporte des paramètres autres que le résultat.
- Retenons qu'une fonction est essentiellement un module de calcul et de renvoi une seule valeur d'un type simple.

5. Remarque :

- Tout nouvel objet utilisé dans une fonction est appelé **objet local**.
- Tout objet déclaré dans le programme principal est appelé **objet global**.
- L'appel d'une fonction peut être effectué au niveau du programme principal ou au niveau d'un module appelant.
- Les paramètres formels qui figurent dans l'en tête de la déclaration de la fonction et correspondent à des variables locales utilisés dans son corps.
- Les paramètres effectifs utilisés dans l'instruction d'appel de la fonction et sont substitués aux paramètres formels au moment de l'appel.
- Evitez d'y insérer les actions d'entrée et de sortie.

III. Les procédures :

1. Activité 2 :

Ecrire un algorithme et son implémentation en Python d'un programme qui permet de saisir 3 entiers a, b et c puis les affichés avec a suivi de a étoiles et b suivi de b étoiles et c suivi de c étoiles.

✧ **Remarque :**

☞ On ne peut pas utiliser des fonctions comme solution car une fonction est essentiellement un module de calcule. Donc on a besoin de définir une autre manière de décomposition.

2. Vocabulaire et syntaxe :

Algorithme
Procédure Nom_procédure (pf1: type1, pf2: type2, … , pfn: typen) **DEBUT** <Traitement> **FIN**

Turbo Pascal
def Nom_procédure (pf1, pf2, … , pfn) **:** <Traitement>

3. Solution de l'activité 2 :

Algorithme	Implémentation en Python
…… …… …… …… …… …… …… ▪ **T.D.O :** **Objet** / **Type/Nature** …… / …… …… …… …… …… …… …… ▪ **T.D.O :** **Objet** / **Type/Nature** …… / ……	…… …… …… …… …… …… …… …… …… …… …… …… …… …… …… ……

4. 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.

5. Les paramètres :

a. Les types de paramètres :

Il existe deux types de paramètres :

- Les paramètres **formels** qui figurent dans l'en tête de la déclaration de la procédure et correspondent à des variables locales utilisés dans son corps.
- Les paramètres **effectifs** utilisés dans l'instruction d'appel de la fonction et sont substitués aux paramètres formels au moment de l'appel.

b. Les modes de passages des paramètres :

La substitution des paramètres effectifs aux paramètres formels s'appelle mode de passage de paramètres. Il existe deux modes de passages de paramètres :

- **Passage par valeur :**
  Dans ce cas, toute modification des paramètres formels n'a aucun effet sur les paramètres effectifs.
  Le transfert de l'information se fait toujours du programme appelant vers le programme appelé.

- **Passage par variable :**
  Dans ce cas, toute modification portés sur le paramètre formel est la même sur le paramètre effectif. Le transfert d'information entre le programme appelant et le programme appelé se fait dans le 2 sens.

c. Remarque :

- Avec les fonctions on utilise toujours le passage de paramètre par valeur.
- Comment choisir le mode de passage de paramètre ?
  - ☞ Si le paramètre est modifié par la procédure (lecture ou affectation) alors on choisit le passage par variable.
  - ☞ Si le paramètre n'est pas modifié par la procédure (affichage ou le paramètre est utilisé comme donné pour faire un traitement quelconque) alors on utilise le passage par valeur.
- Si le mode de passage est par variable (par référence, par adresse), on ajoutera le symbole **@** avant le nom du paramètre.