



python



Montplaisir Imm.
Espace Tunis Bloc H.
2ème étage. Bureau 6



FORMATION PYTHON

- 1 L'essentiel du cours**
- 2 Exercices Algorithme**
- 3 Exercices Python**
- 4 Astuces et Remarques**

Centre Formakt_Bac

Contact:

+216-55-530-162

formakt.bac@gmail.com



Avant de commencer, faisons une ample connaissance

Qui sommes-nous ?

Nous sommes **Formakt-Bac**, un centre spécialisé dans l'enseignement de la matière informatique pour les lycéens et les collégiens. Nous proposons des cours en présentiel ou à distance. La formation avec nous a un autre goût, car nous travaillons à simplifier l'information et à la communiquer facilement à l'élève.

Nos élèves témoignent de notre efficacité, plus de 94% de nos participants ont développé leurs compétences en informatique et ont réussi à obtenir d'excellentes notes.



Allez ! attachez vos ceintures, c'est l'heure d'un voyage pédagogique ludique

Python c'est quoi ?

Python est un **langage de programmation** inventé par **Guido van Rossum**. La première version de python est sortie en 1991.

Python est un langage de programmation interprété, c'est-à-dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter. Si vous avez touché un peu la programmation, vous verrez que ce langage possède une certaine poésie. Les programmeurs s'amusent souvent à trouver la manière la plus jolie/efficace d'écrire une suite d'instruction. Rares sont ceux qui critiquent la logique Python contrairement à **JavaScript** par exemple.

Que fait Python ?

Python est à la fois simple et puissant, il vous permet d'écrire des scripts très simples mais grâce à ses nombreuses bibliothèques, vous pouvez travailler sur des projets plus ambitieux.

*Web : Aujourd'hui **Python** combiné avec le framework **Django** est un très bon choix technologique pour des gros projets de sites internet.

*Système : **Python** est également souvent utilisé par les admins système pour créer des tâches dites répétitives ou simplement de maintenance. D'ailleurs si vous voulez créer des applications java en codant en **Python**, c'est possible grâce au projet **Jython**.

Pourquoi préférer Python aux autres langages ?

Python est un langage facile à apprendre et son code est plus lisible, il est donc plus facile à maintenir. Il est parfois jusqu'à 5 fois plus concis que langage **Java** par exemple, ce qui augmente la productivité du développeur et réduit mécaniquement le nombre de bugs.

Python est également utilisé dans les milieux scientifiques, par exemple la **bioinformatique**. Des librairies sont disponibles pour ce domaine comme le module **biopython**.

Il existe également des bibliothèques facilitant la création de jeux vidéo en 2D et 3D, exemple **pyGame**.

Qui utilise Python ?

Google (Guido van Rossum a travaillé pour Google de 2005 à 2012), **Yahoo**, **Microsoft**, La **Nasa** revendique l'utilisation de python...

Sommaire du livre

1. Chapitre 1 :Les structures simples.....7

Sont les premiers pas chers élèves. A travers ce chapitre, vous apprenez les bases de la programmation, les commandes de saisi, d'affichage, d'affectation... grâce à un certain nombre d'applications et d'astuces importantes. Ne terminez pas ce chapitre sans en profiter, nous sommes de votre côté.

Voici une idée de ce que contiendra le chapitre

1.1.	Affichage	7
1.2.	Affectation	7
1.3.	Saisie de donnée	8
	Exercice 1: Somme de deux chiffres	9
	Correction de l'exercice	10
	Exercice : Somme de trois chiffres	12
	Correction de l'exercice	13
1.4.	Les fonctions sur les types numériques	14

2. Chapitre 2 : Les structures de contrôle conditionnelles16

Comment s'est passée votre expérience avec le premier chapitre ? Êtes-vous prêt pour une nouvelle expérience ? Au fur et à mesure que nous parcourrons ce chapitre, nous apprendrons les structures de contrôle conditionnelles, comment différencier entre les trois formes et comment les appliquer. Nous sommes encore ensemble.

Et c'est ce que ce chapitre contiendra

2.1.	Forme simple	16
	Exercice : La racine carrée	16
	Correction de l'exercice	16
	Exercice : Parité	18
	Correction de l'exercice	18
2.2.	Forme alternative	20
	Exercice : Parité	20

Correction de l'exercice	20
2.3. Forme généralisée	21
Exercice : Moyenne	22
Correction de l'exercice	22
Exercice : Nombre Cubique	23
Correction de l'exercice	23

3. Chapitre3 : Chaîne de caractères et ses méthodes25

Nos chers élèves, êtes-vous prêt pour une aventure plus avancée ? Ce chapitre est la clé pour maîtriser les chaînes de caractères avec tous leurs problèmes. Nous vous avons préparé un tableau contenant toutes les fonctions prédéfinis et un certain nombre d'applications qui vous permettront de résoudre la plupart des exercices qui vous seront posés.

Le chapitre sera divisé comme suit

3.1. Le type Caractère	25
3.2. Définition d'une chaîne	25
3.3. Extraction d'une sous chaîne	25
3.4. Concaténation et multiplication	26
3.5. Les méthodes	26
Exercice : E-mail	28
Correction de l'exercice	29
Exercice : Nombre magique	30
Correction de l'exercice	31

4. Chapitre4 :Les structures de contrôle itératives33

Puisque vous êtes avec nous maintenant, nous ne vous craindrons pas. Préparez-vous à plus de suspense. Ce chapitre contient les parties les plus importantes pour le développement d'un programme en informatique " les structures de contrôles itératives". Nous commencerons par l'étude de la fameuse boucle "Pour", beaucoup de plaisir et d'exercices vous attendent, la plupart des exercices typiques sont à votre disposition dès maintenant.

Lorsque nous serons sûrs que vous maîtriserez tout ce qui précède, nous travaillerons ensemble pour étudier les boucles "Répéter" et "Tant que"

Don't worry ! vous pourrez faire la distinction entre ces 2 boucles et la boucle "Pour" et le domaine d'application de chacune.

Ce chapitre contiendra les paragraphes suivants

4.1.	Boucle Pour	33
4.2.	Notion du compteur	34
	Exercice : Somme de 5 entiers	35
	Correction de l'exercice	37
	Exercice : Somme des chiffres	38
	Correction de l'exercice	38
	Exercice : Somme des chiffres	39
	Correction de l'exercice	39
	Exercice : Nombre des espaces	41
	Correction de l'exercice	41
4.3.	Modification de la valeur d'une chaîne de caractères	42
	Exercice : Remplacer les caractères	43
	Correction de l'exercice	43
	Exercice : Concaténation	44
	Correction de l'exercice	44
4.3.1.	Autre méthode de la boucle for	46
	Former une chaîne (copie) de la chaîne saisie	46
	Exercice : Modification chaîne de caractères	47
	Correction de l'exercice	47
	Exercice : Modification chaîne de caractères	50
	Correction de l'exercice	50
	Exercice d'application	51
	Transformer une chaîne en majuscule	52

Changer les caractères d'une chaîne donnée par leurs codes ASCII	52
Exercice : Factorielle	52
Correction de l'exercice	52
Exercice : Diviseurs	55
Correction de l'exercice	56
Exercice : Afficher les diviseurs	56
Correction de l'exercice	56
Exercice : Somme des diviseurs	57
Correction de l'exercice	58
Exercice : Nombre des diviseurs	59
Correction de l'exercice	59
Exercice Nombre Premier	60
Correction de l'exercice	60
4.4. Boucle Tant que	62
4.5. Boucle répéter	63
Exercice PGCD	68
Correction de l'exercice avec la boucle répéter	69
Correction de l'exercice avec la boucle tant que	70
Exercice nombre parfait	71
Correction de l'exercice	72
Exercice Palindrome	74
Correction de l'exercice avec la boucle tant que	77
Exercice Etoiles	80
5. Chapitre5 : Les Tableaux	86

L'équipe Formakt-Bac vous encourage pour cet effort. Nous sommes très fiers de vous car vous avez continué avec détermination à vous approcher du sommet. Il est donc temps de vous familiariser avec les tableaux dont ils vous parlent toujours. Nous vous présenterons les

manières de remplir, d'afficher, de trier, de parcourir et de traiter les tableaux. Et comme d'habitude Formakt-Bac vous prouvera que la programmation informatique est facile avec nous.

5.1.	Définition	86
5.2.	Utilisation.....	86
5.3.	Opération sur les listes	87
5.4.	Remplissage d'un tableau	89
5.5.	Autre méthode du remplissage du tableau	90
	Remplissage d'un tableau (avec la bibliothèque numpy)	91
	Exercice : Somme des carrées des chiffres	93
	Correction de l'exercice	93
5.6.	Le maximum / minimum dans un tableau.....	95
5.7.	Exercice longueur maximale des chaînes.....	96
	Exercice : Tableau des chaînes	96
	Correction de l'exercice	96
6.	Chapitre 6 : Sous-Programme.....	98

Houlla !! Félicitations pour avoir atteint ce stage. Apportez un bloc note et des stylos, Lisez bien ce chapitre. Relisez-le. Extrayez la différence entre "La procédure" et "La fonction", "Les paramètres formels" et "Les paramètres effectifs", "Les variables globales" et "Les variables locales" et pratiquez autant que possible.

La planification de votre dernier voyage sera la suivante :

6.1.	Syntaxe des sous-programmes	98
6.1.1.	Procédure.....	98
6.1.2.	Fonction.....	100

7. Chapitre 7 : Les algorithmes de tri.....109

Tant que vous êtes ici, cela signifie que vous êtes un élève du baccalauréat. Vous êtes sur le point de terminer votre parcours avec la matière informatique.

Dans ce chapitre, nous vous présenterons les deux algorithmes de tri qui vous intéressent : le tri par sélection et le tri à bulles. En plus la correction de deux sujets de bac

7.1.	Introduction	109
7.2.	Les algorithmes de tri	109
7.2.1.	Tri par sélection	109
7.2.2.	Tri à bulles	115
	Exercice d'application : Bac pratique 2009	118
	Exercice d'application : Bac théorique 2013	125

8. Chapitre 8 : Les algorithmes de recherche128

Une fois que vous êtes en mesure de trier les tableaux, vous pouvez maintenant familiariser avec certains algorithmes de recherche d'un élément dans un tableau.

Dans ce chapitre, nous allons découvrir deux algorithmes de recherche, le premier pour une recherche séquentielle et le second pour une recherche dichotomique

8.1.	Introduction	128
8.1.1.	Recherche séquentielle	128
8.1.2.	Recherche dichotomique	129

9. Chapitre 9 : Les interfaces graphiques131

Nous sommes en train de finaliser ce voyage, c'est le dernier chapitre de ce livre. Au cours de ce chapitre nous apprendrons à connaître l'outil à utiliser pour concevoir nos interfaces graphiques et les modifications et les ajouts nécessaire dans les codes python à développer

9.1.	Introduction	131
9.2.	Qt designer + PyQt5	131
9.3.	Utilisation de l'outil Qt Designer	134
9.4.	Partie programmation	135
9.5.	Exemple pratique	136

1. Chapitre 1 :Les structures simples

1.1. Affichage

Si nous voulons afficher un message, nous utilisons les instructions suivantes :

Algorithme	Python
Ecrire ('Le message à afficher')	Print ('Le message à afficher')

Exemples :

Ecrire('python')

print (' python ') → le programme donne : python

Pour renvoyer deux messages écrits avec des commandes print() distinctes sur la même ligne, on utilise le syntaxe suivant :

```
print (' formation ', end = '')
```

```
print (' python ')
```

Dans ce cas le programme donne → formationpython

1.2. Affectation

Si on veut créer une variable x de type entier et on veut lui donner la valeur 2, on utilise les syntaxes données par le tableau suivant :

Algorithme	Python
x ← 2 (Se lit x reçoit 2)	x=2

Cette opération est dite affectation : C'est-à-dire mettre la valeur 2 dans la variable x (on dit x reçoit 2 ou x prend la valeur 2).

Pour afficher un message, il faut l'écrire entre deux côtes "", en informatique il y'a différence d'écrire : Ecrire(x) et Ecrire ('x') ainsi que print(x) et print('x')

Dans le premier cas (sans apostrophes) signifie qu'on veut afficher le contenu de la variable x donc le programme affiche 2 alors qu'avec les deux apostrophes le programme affichera la lettre x tel qu'elle est écrite.

1.3. Saisie de donnée

Pour donner l'accès à l'utilisateur de saisir des données par le clavier, on doit utiliser les instructions suivantes :

Algorithme	Python
lire(a)	a=input()

La valeur saisie par l'utilisateur sera enregistrée dans la variable a.



Généralement, lorsqu'on demande à l'utilisateur de saisir des données, on lui affiche un message qui porte des informations sur ce qu'on attend de

Exemples :

Algorithme	Python
Ecrire ('Donner ton prénom') lire(p)	print("Donner ton prénom") p=input()

Python nous a permis de diminuer le nombre des lignes et de combiner l'affichage du message avec le saisi du donné.

Et donc ça sera : p=input ('Donner ton prénom') au lieu d'écrire : print ("Donner un entier")

n = input ()

Exercice 1: Somme de deux chiffres

Ecrire un programme qui permet de saisir un entier composé de 2 chiffres et calculer la somme de ses chiffres.

Exemple l'utilisateur saisi l'entier 23 le programme affiche 5 (2+3)

L'idée est de séparer le chiffre d'unité du chiffre de dizaine.

Il est évident de penser à utiliser la division euclidienne de l'entier donné par l'utilisateur par 10.

$$\begin{array}{r}
 23 \quad | \quad 7 \\
 2 \quad | \quad 3 \\
 \text{reste} \qquad \text{quotient} \\
 \downarrow \qquad \qquad \downarrow \\
 23 \bmod 7 \quad 23 \text{ div } 7 \\
 23 \% 7 \quad 23 // 7
 \end{array}$$



Avec une division euclidienne sur 10, le nombre d'unités présente le reste de la division alors que le nombre de dizaine présente le quotient

Pour obtenir le reste et le quotient en informatique, on utilise les commandes suivantes :

Algorithme	Python
Quotient : $x \text{ div } 10$	$x // 10$
Reste : $x \text{ mod } 10$	$x \% 10$

Correction de l'exercice

Algorithme :

Debut Chiffres2

Ecrire ('donner un entier composé de deux chiffres')

Lire (n)

$u \leftarrow n \bmod 10$

$d \leftarrow n \text{ div } 10$

$s \leftarrow u+d$

Ecrire (s)

Fin Chiffres2

Objets	type/Nature	Rôle
n	Entier	Dozané
u	Entier	Chiffre d'unités
d	Entier	Chiffre de dizaine
s	Entier	La somme des deux chiffres

Python

```
1 n=int(input('donner un entier composé de deux chiffres'))
2 u=n%10
3 d=n//10
4 s=u+d
5 print(s)
```

Console

Python 3.7.9 (bundled)

>>> %Run Cours.py

donner un entier composé de deux chiffres

23

>>>



Mais pourquoi nous avons écrire int avant la commande input() ???

C'est parce que la fonction input retourne par défaut une chaîne de caractères

Donc, on doit convertir la valeur saisie en type entier pour qu'on puisse appliquer par la suite les opérations nécessaires (la somme et la division)

Voir cet exemple

```
1 n = input ('Donner un entier')
2 s = 5 + n
3 print (s)
```

Console

Donner un entier

5

Traceback (most recent call last):

File "C:\Users\ASUS\Desktop\Bureau\Formakt Bac\Bac Français\t608.py", line 2, in <module>

 s = 5 + n

TypeError: unsupported operand type(s) for +: 'int' and 'str'

13

Le programme affiche un message d'erreur : car on ne peut pas sommer une chaîne de caractère à un entier.

```
1 n = int (input ('Donner un entier'))
2 s = 5 + n
3 print (s)
```

```
Console:
>>> %Run t608.py
Donner un entier5
10
```

Après conversion de la variable `n` en un entier le programme tourne normalement sans aucun problème et affiche le résultat souhaité.



L'affichage Mixte : On peut afficher à la fois un message et le contenu d'une variable en séparant entre eux par une virgule.

Exemple : `a ← 18`

```
Ecrire('Bonjour j ai ',a,' ans ')
```

Le programme affiche Bonjour j'ai 18 ans.

Dans le cas de notre exercice on peut écrire :

```
1 #Exercice : Saisir un entier composé de 2 chiffres
2 #et calculer la somme de ses chiffres
3 n = int (input ('Donner un entier composé de 2 chiffres'))
4 d = n // 10
5 u = n % 10
6 s = d + u
7 print ('La somme des chiffres',d,'et',u,'est',s)
```

```
Console:
>>> %Run t608.py
Donner un entier composé de 2 chiffres72
La somme des chiffres 7 et 2 est 9
```

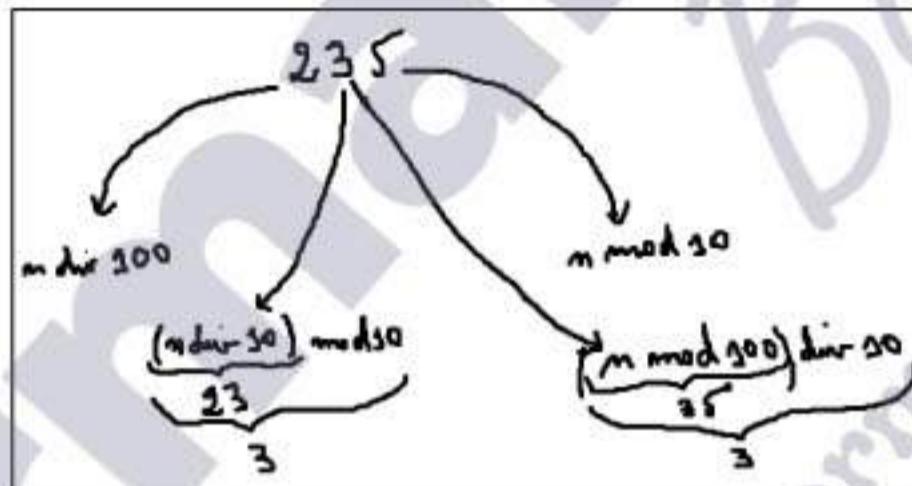
Exercice : Somme de trois chiffres

Ecrire un programme qui permet de saisir un entier composé de 3 chiffres et calculer la somme de ses chiffres

Exemple l'utilisateur saisi l'entier 235 le programme affiche 10 (2+3+5)



Nous utiliserons le même principe que l'exercice précédent mais avec une division euclidienne sur 100 au lieu de 10.



Correction de l'exercice

Algorithme :

Début Somme_chiffres

Ecrire ("Donner un entier composé de 3 chiffres")

Lire (n)

$c \leftarrow n \text{ div } 100$

$d \leftarrow (n \text{ div } 10) \text{ mod } 10$

$u \leftarrow n \text{ mod } 10$

$s \leftarrow c + d + u$

Ecrire ("La somme des chiffres", c, " ", d, "et", u, "est", s)

Fin Somme_chiffres

Objet	Type/Nature	Rôle
n	entier	donnée
c	entier	centaine
d	entier	dizaine
u	entier	unité
s	entier	Somme des chiffres

Python:

```

1 #Exercice : Saisir un entier composé de 3 chiffres
2 #et calculer la somme de ses chiffres
3 n = int (input ('Donner un entier composé de 3 chiffres'))
4 c = n // 100
5 d = (n // 10) % 10
6 u = n % 10
7 s = c + d + u
8 print ('La somme des chiffres',c,',',d,'et',u,'est',s)

```

Console

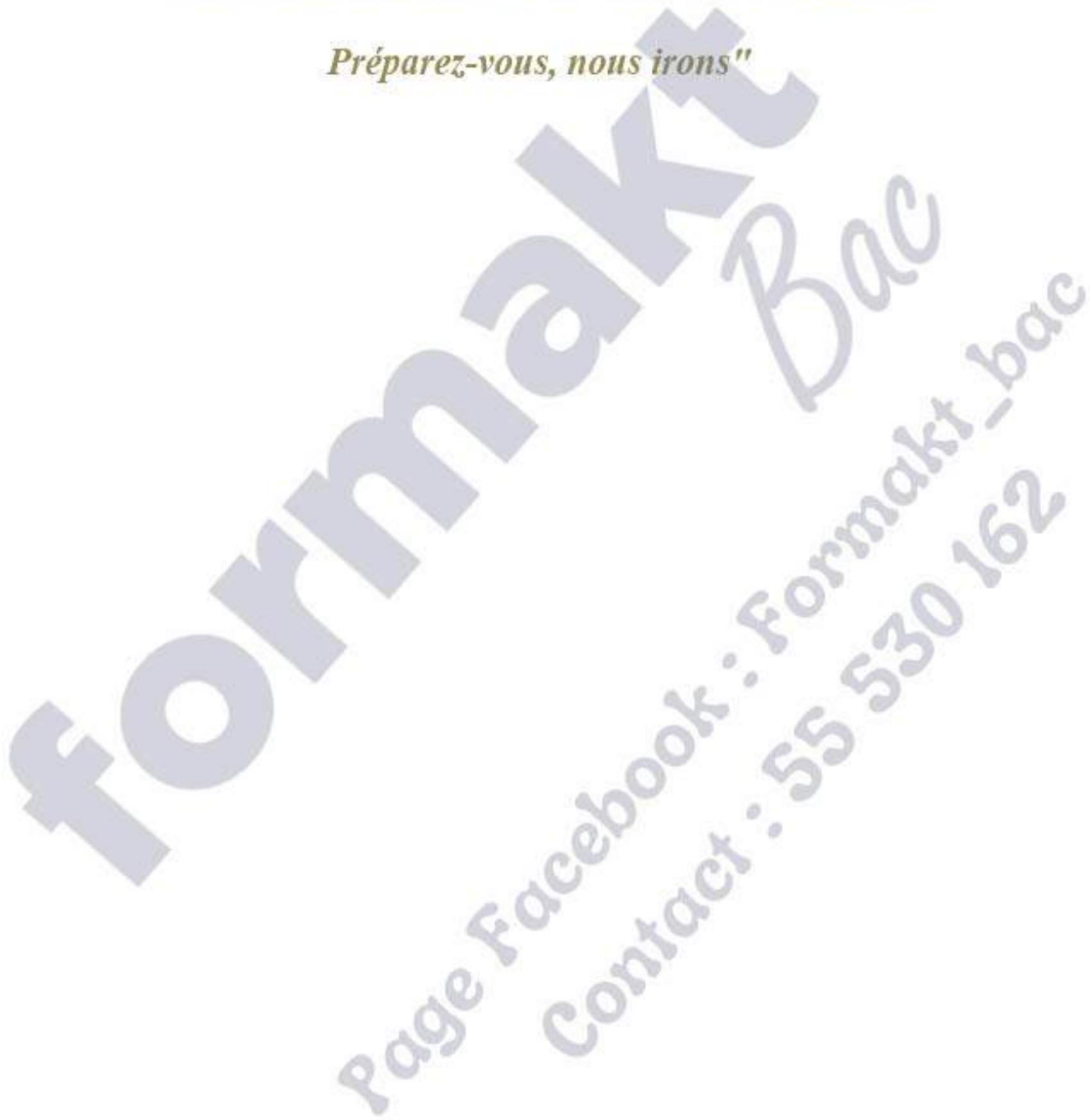
>>> %Run t608.py

Donner un entier composé de 2 chiffres235
La somme des chiffres 2 , 3 et 5 est 10**1.4. Les fonctions sur les types numériques :**

Algorithme	Python	Rôle	Exemple
Racine_carrée(x)	sqrt(x)	Donne la racine carrée d'un réel x	from math import * x = 16 sqrt(x) → donne 14.0
abs(x)	abs(x)	Donne la valeur absolue d'un réel x	x = -2 abs(x) → donne 2
Tronc(x)	trunc(x)	Enlève la partie décimale d'un réel x	from math import * x = 2.96 trunc(x) → donne 2
Arrondi(x)	round(x)	Donne l'entier le plus proche du réel x	x = 2.5 → donne 2 x = 2.51 → donne 3
pow(x,y)	pow(x,y)	Donne x puissance y	pow (2,3) → 8
Aléa (x,y)	randint(x,y)	Donne un entier aléatoire entre x et y	from random import * x = randint (0,10) donne → un entier entre 0 et 10
Aléa (x,y)	uniform (x,y)	Donne un réel aléatoire entre x et y	from random import * x = uniform (0,10) donne → un réel entre 0 et 10

"Ce chapitre est terminé, vous a-t-il été utile ? Je sais que
vous attendez le prochain chapitre avec impatience

Préparez-vous, nous irons"



2. Chapitre 2 : Les structures de contrôle conditionnelles

Ces structures sont utilisées lorsque le programmeur souhaite de vérifier la validation de certaines conditions nécessaires pour le travail.

2.1. Forme simple

Algorithme	Python
Si Condition alors Résultat Fin Si	if Condition : <i>Résultat</i>

Exercice : La racine carrée

Ecrire un programme qui permet de faire calculer la racine carrée d'un réel x

Correction de l'exercice

Algorithme :

Début racine

Ecrire ("Donner un réel ")

Lire(x)

Si $x \geq 0$ alors écrire("La racine carrée est =", racine_carrée(x))

Fin Si

Fin racine

TDO

Objet	Type/Nature	Rôle
x	Réel	Donné

Python :

```

1 x = float (input('Donner un réel'))
2 if x >= 0 :
3     print ('La racine carrée est',sqrt(x))
4

```

Donner un réel16
Traceback (most recent call last):
File "C:\Users\ASUS\Desktop\Bureau\Formakt Bac\Bac Français\t608.py", line 3, in <module>
 print ('La racine carrée est',sqrt(x))
NameError: name 'sqrt' is not defined



C'est quoi le 'sqrt' ??

sqrt est une fonction prédéfinie dans python qui nous a permis de calculer la racine carrée d'un réel.

Si elle est prédéfinie pourquoi ne la connaît-il pas ?

→ Pour y accéder, vous devez d'abord importer la bibliothèque math.

```

1 from math import*
2 x = float (input('Donner un réel'))
3 if x >= 0 :
4     print ('La racine carrée est',sqrt(x))
5

```

Donner un réel16
La racine carrée est 4.0



Lorsque vous appuyez sur Entrée après les deux points, vous remarquerez que le curseur se déplace vers la droite. Cet espace est appelé indentation, tant que l'indentation est présente, nous restons dans le résultat de if.

```

2 if x >= 0 :
3     print ('La racine carrée est',sqrt(x))
4

```

↓
indentation

Exercice : Parité

Ecrire un programme qui permet de vérifier la parité d'un entier (en utilisant la forme simple)

Correction de l'exercice

Algorithme :

Début parité

Ecrire ("Donner un entier ")

Lire(n)

Si $x \bmod 2 = 0$ alors Ecrire(n, 'est pair')

Fin Si

Si $x \bmod 2 \neq 0$ alors Ecrire(n, 'est impair')

Fin Si

Fin parité

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné

Python :

```

1 n = int (input ('Donner un entier à vérifier'))
2 if n % 2 == 0 :
3     print (n,'est pair')
4 if n % 2 != 0 :
5     print (n,'est impair')

```

Les symboles de comparaison :

Les opérateur	Syntaxe en algorithme	Syntaxe en python
L'addition	(+)	+
La soustraction	(-)	-
La multiplication	(*)	*
La division	(/)	/
La division entière	Div	//
Le reste de la division entière	Mod	%
Strictement supérieur	>	>

Strictement inférieur	<	<
Supérieur ou égale	\geq	\geq
Inférieur ou égale	\leq	\leq
Affectation	\leftarrow	=
Different	\neq	\neq
La comparaison	=	\equiv
Et	Et	And
Ou	Ou	Or

2^{ème} méthode de correction

Début parité

Ecrire("Donner un entier ")

Lire (n)

message \leftarrow "Pair"

Si $n \bmod 2 \neq 0$ alors

message \leftarrow "impair"

FinSi

Ecrire(n, " est ", message)

Fin parité

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
message	Chaine de caractères	message qui contient l'information sur la parité de n

Python :

```

1 n = int (input ('Donner un entier à vérifier'))
2 message = 'pair'
3 if n % 2 != 0 :
4     message = 'imapir'
5 print (n , 'est', message)

```

2.2. Forme alternative

Algorithme	Python
Si Condition alors Résultat 1 Sinon Résultat 2 Fin Si	if Condition : Résultat1 else : Résultat 2

Exemple

Si $x \geq 0$ alors écrire("Positif")

Sinon écrire ("négatif")

Fin Si

Exercice : Parité

Ecrire un programme qui permet de vérifier la parité d'un entier (en utilisant la forme alternative)

Correction de l'exercice

Algorithme :

Début parité

Ecrire ("Donner un entier ")

Lire(n)

Si $x \bmod 2 = 0$ alors Ecrire(n, 'est pair')

Sinon Ecrire(n, 'est impair')

Fin Si

Fin parité

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné

Python :

```

1 n = int(input('Donner un entier à vérifier'))
2 if n % 2 == 0 :
3     print(n, 'est pair')
4 else :
5     print(n, 'est impair')

```

2.3. Forme généralisée

Algorithme	Python
Si Condition alors Résultat 1	if Condition :
Sinon si Condition 2 alors Résultat 2	Résultat 1
Sinon Si Condition 3 alors Résultat 3	elif condition 2 :
	Résultat 2
	elif condition 3 :
	Résultat 3
Sinon si Condition n-1 alors Résultat n-1	
Sinon Résultat n	
Fin Si	
	elif condition n-1 :
	Résultat n-1
	else :
	Résultat n

On n'écrit jamais une condition avec sinon (else)

Exercice : Moyenne

Ecrire un programme qui permet de saisir une moyenne (moy) puis affiche la décision correspondante :

ADMIS si $moy \geq 10$

CONTROLE si $9 \leq moy < 10$

REDOUBLE si $moy < 9$

Correction de l'exercice

Algorithme :

```
Début décision
Ecrire("Donner une moyenne")
Lire(moy)
Si moy>=10 alors message ← "ADMIS"
Sinon Si moy>=9 alors message ← "CONTROLE"
Sinon message ← "REDOUBLE"
Fin Si
Ecrire(message)
Fin décision
```

TDO

Objet	Type/Nature	Rôle
moy	Réel	Donné
message	Chaine de caractères	Message qui contient la décision

Python :

```
1 moy = float (input('Donner une moyenne'))
2 if moy >= 10 :
3     message = 'ADMIS'
4 elif moy >= 9 :
5     message = 'CONTROLE'
6 else :
7     message = 'REDOUBLE'
8 print (message)
```

Exercice : Nombre Cubique

Ecrire un programme qui permet de saisir un entier n contenant 3 chiffres puis affiche si le nombre est cubique ou non.

Un nombre est dit cubique s'il est égal à la somme des cubes de ses chiffres.

Exemple 153 est cubique puisque $1^3 + 5^3 + 3^3 = 153$

Correction de l'exercice

Algorithme :

Début cubique

Écrire("Donner un entier composé de 3 chiffres")

Lire(n)

c \leftarrow n div 100

d \leftarrow (n div 100) mod 10

u \leftarrow n mod 10

s \leftarrow c³ + d³ + u³

Si s=n alors Ecrire (n, 'Cubique ')

Sinon Ecrire(n, ' n est pas cubique ')

Fin Si

Fin cubique

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
c	Entier	le nombre de centaine de n
d	Entier	le nombre de dizaine de n
u	Entier	le nombre d'unités de n
s	Entier	la somme des cubes des chiffres de n

Python :

```
1 n = int (input('Donner un entier composé de 3 chiffres'))
2 c = n // 100
3 d = (n // 10) % 10
4 u = n % 10
5 s = c**3 + d**3 + u**3
6 if s == n :
7     print (n, 'est cubique')
8 else :
9     print (n, 'n est pas cubique')
```

Avez-vous remarqué !! un autre chapitre est terminé. Nous approchons rapidement du sommet

Continuons alors !

3. Chapitre3 : Chaine de caractères et ses méthodes

3.1. Le type Caractère

On a 256 caractères (255 (clavier) + " vide)

Chaque caractère a un code ASCII (American Standard Code for Information Interchange)

$$'A' \rightarrow 65 \quad 'a' \rightarrow 97$$

On peut comparer les caractères

`'A' < 'a'`, car le code ASCII de 'A' est inférieur au code ASCII de 'a'

3.2. Définition d'une chaîne

```
ch = 'python'
ch = "python"
ch="python"
ch = " " "python" "
ch = '' ou ch = str() → chaîne vide
ch[1] → donne 'y'
```

L'outil `ch[i]` tel que les `i` sont indices des caractères numérotés de 0 à `long(ch) - 1`, nous permet d'accéder en mode lecture à chaque caractère de la chaîne

`ch[-1] → donne le dernier caractère 'n'`

3.3. Extraction d'une sous chaîne

```
ch[id : if]
id : indice de début
if : indice de fin - 1
```

```

ch = 'python'

ch[0 : 3] → donne 'pyt'

ch[2 :] → donne 'thon'

ch[-2 :] → donne 'on'

ch[:-2] → donne 'pto'

ch[:: -2] → donne 'nhy'

```

3.4. Concaténation et multiplication

"formation"+ "python" → donne "formationpython"

Ch = "a" * 5 → donne ch = "aaaaa"

3.5. Les méthodes

ord(x)	Code ASCII du caractère x	Ord ('A') → donne 65 Ord ('a') → donne 97
chr(x)	Caractère dont le code ASCII est x	Chr(65) → 'A' Chr(97) → 'a'
len (ch)	Taille de la chaîne	Len ('python') → donne 6
int (ch)	Convertir une chaîne en un nombre entier	Int('120') → donne 120
float(ch)	Convertir une chaîne en un nombre réel	int('12.50') → donne 12.50
str (objet)	Convertir l'objet en une chaîne	Str (120) → donne '120' Str (12.50) → donne '12.50'
max (ch)	Caractère ayant le code ASCII le plus élevé ou le plus faible	max ('python') → donne 'y'
min (ch)		
ch.upper()	Convertir ch en majuscule	'python'.upper() → donne 'PYTHON'
ch.lower()	Convertir ch en minuscule	'PYTHON'.lower() → donne 'python'
ch.capitalize()	Convertir le 1 ^{er} caractère de ch en majuscule	'python'.capitalize() → donne 'Python'
ch.title()	Convertir le 1 ^{er} caractère de chaque	'formation python'.title() → donne

	mot de ch en majuscule	'Formation Python'
ch.center(n,ch)	Centrer ch dans une chaîne de n caractère car	'python'.center(10,'*') → donne ***python***
Ch.replace(ch1 ,ch2,n)	Remplace tous le n occurrence de ch1 par ch2 dans la chaîne ch	'informatique'.replace('i','I') → donne 'InformatIque' 'informatique'.replace('i','I',1) → donne 'Informatique'
ch.count(ch1,d,f)	Compter le nombre d'occurrence de ch1 dans ch dans un intervalle des caractères [d,f-1]	'informatique'.count('i') → donne 2 'informatique'.count('i',0,4) → donne 1
ch.find(ch1,d,f)	Retourne la 1ère position de ch1 dans une portion de caractère de ch	'informatique'.find ('i') → donne 0 'informatique'.find ('i',1,10) → donne 8 'informatique'.find ('I') → donne -1
ch.strip()	Enlever les espaces avant et après	' python '.strip → donne 'python'
Méthodes de vérification	ch.isupper(), ch.islower(), ch.isalpha(), ch.isalnum(), ch.isdecimal() : sont des fonctions booléens	'12.20'.isdecimal() → donne faux 'python'.isalpha() → donne vrai 'PYTHON'.isupper() → donne vrai 'Python'.islower() → donne faux 'PYTHON3x'.isalnum() → donne vrai

Algorithme	Code en Python	Rôle	Exemples
Concat (ch1, ch3,...,chn) <u>ou</u> Ch1 + ch2 + ... +chn	Ch1 + ch2 + ... + chn	Permet la concaténation d'un ensemble de chaînes de caractères	Ch1 = '2020' Ch2 = '/' Ch3='2021' Ch = concat(ch1, ch2, ch3) '2020/2021'
Pos(ch1,ch2)	Ch2.find(ch1)	Retourne la première position de la chaîne ch1 dans la chaîne ch2	Ch1='2' Ch2 = '2020' Pos (ch1,ch2) = 0
Convch (n)	Ch = str (n)	Convertir le nombre n en une chaîne de caractères	Ch = Convch(2020) Ch = '2020'

Valeur (ch)	n = int (ch)	Convertir une chaîne de caractère en une valeur numérique (sinon erreur)	ch = '2020' n = valeur (ch) n = 2020 Str (12.50) → donne '12.50'
	n = float (ch) si ch contient un réel		Ch= '2info2' n = valeur (ch) n = erreur
Sous_chaine (ch,id,if)	Ch [id :if] id : indice de début if : indice de fin -1	Retourne une partie de la chaîne ch à partir de la position id jusqu'à la position if -1	ch = 'informatique' ch2 = ch[2:5] print(ch2) le programme affiche : for
Effacer (ch, d, f)	Ch[:d]+ch[f:]	Efface des caractères de la chaîne ch à partir de la position d jusqu'à la position f (f exclue).	ch = 'informatique' ch2 = ch[:4] + ch[7:] print (ch2) le programme affiche : infotique

Exercice : E-mail

Ecrire un programme qui permet de saisir une adresse mail de la façon suivante : prénom.nom@serveur.extension puis affiche chaque partie seule.

Exemple : formakt.bac@gmail.com

Prénom : formakt

Nom : bac

Serveur : gmail

Extension : com

```

1 ch = input ('Donner un email')
2 prenom = ch[0:3]
3 print (prenom)

```

Console :
Donner un email
yea

Fautes à éviter !!!!

Le programme doit être exécutable pour n'importe quelle personne.



L'idée est d'utiliser les positions du caractère « . » et « @ », c'est claire que quel que soit le prénom de l'utilisateur de notre programme commence dès le début de l'e-mail jusqu'à la position du caractère « . »

Correction de l'exerciceAlgorithme :

```

Début email
Ecrire("Donner un email")
Lire(ch)
prenom ← sous_chaine(ch,0,pos('.',ch))
Ecrire('Prénom :',prenom)
nom ← sous_chaine(ch,pos('.',ch)+1,pos('@',ch))
Ecrire('Nom :',nom)
Effacer (ch, 0, pos('@',ch)+1)
serveur ← sous_chaine(ch,0, pos('.',ch))
Ecrire('Serveur :',serveur)
extension ← sous_chaine(ch,pos('.',ch)+1, long(ch))
Ecrire ('Extension : ',extension)
Fin email

```

TDO

Objet	Type/Nature	Rôle
ch	chaine de caractères	Donnée
prenom	chaine de caractères	sous chaine de l'e-mail qui contient le prénom
nom	Chaine de caractères	sous chaine de l'e-mail qui contient le nom
serveur	chaine de caractères	sous chaine de l'e-mail qui contient le serveur
extension	chaines de caractères	sous chaine de l'e-mail qui contient l'extension

Python :

```

1 ch = input ('Donner un email')
2 prenom = ch[ 0 : ch.find('.') ]
3 print (prenom)
4 nom = ch[ ch.find(',') +1 : ch.find('@') ]
5 print (nom)
6 ch2 = ch[:0] + ch[ch.find('@')+1 :]
7 print (ch2)
8 serveur = ch2[ 0 : ch2.find('.') ]
9 print (serveur)
10 extension = ch2 [ch2.find(',')+1 : len(ch2) ]
11 print (extension)
12
Console:
Donner un emailformakt.bac@gmail.com
formakt
bac
gmail.com
gmail
com

```

Exercice : Nombre magique

Ecrire un programme qui permet de saisir un entier n contenant minimum 4 chiffres et qui affiche s'il est magique ou non.

Un nombre est dit magique si la somme des deux premiers chiffres est égal au produit de deux derniers chiffres.

Exemple : n=32451, n est magique car $3+2 = 5*1$

Fautes à éviter !!!!

```

1 n = int (input ('Donner un entier'))
2 ch = str(n)
3 s = ch[0] + ch[1]
4 p = ch[2] *| ch[3]
5 if s == p :
6     print (n,'est magique')
7 else :
8     print (n,'est non magique')

```

L'entier `n` est composé au minimum de 4 chiffres, c'est-à-dire il peut être composé de 4 chiffres exactement ou de plus que 4 chiffres. Donc le programme doit être exécutable pour tous les cas.

En plus la variable `s` est de type `str`, ainsi que la variable `p` va poser un problème de syntaxe puisque on ne peut pas multiplier un caractère par un autre caractère.

Correction de l'exercice

Algorithme :

Début magique

Écrire("Donner un entier")

Lire(`n`)

`ch` ← convch(`n`)

`s` ← valeur(`ch[0]`)+valeur(`ch[1]`)

`p` ← valeur (`ch[long(ch)-2]`)* valeur(`ch[long(ch)-1]`)

si `p=s` alors Ecrire(`n`, 'est magique')

sinon Ecrire (`n`, 'est non magique')

Fin Si

Fin magique

TDO

Objet	Type/Nature	Rôle
<code>n</code>	Entier	Donné
<code>ch</code>	Chaine de caractères	

Python :

```

1 n = int (input ('Donner un entier'))
2 ch = str(n)
3 s = int(ch[0]) + int(ch[1])
4 p = int(ch[len(ch)-2]) * int(ch[len(ch)-1])
5 if s == p :
6     print (n,'est magique')
7 else :
8     print (n,'est non magique')

Console:
Donner un entier32451
32451 est magique

```

"Le chapitre des chaînes de caractères est terminé. Essayez de répéter tous les exercices.

Je sais que la quantité d'informations est importante, mais vous êtes capable ! "

4. Chapitre4 : Les structures de contrôle itératives

4.1. Boucle Pour

Il s'agit d'une structure itérative utilisée lorsqu'il y'a une répétition d'une suite d'instructions d'un nombre fini de fois connu à l'avance.

Algorithme	Python
Pour i de 0 à n-1 faire	for i in range (n) :
<i>Traitements à répéter</i>	<i>Traitements à répéter</i>
Fin Pour	

→ Exemple

```

1 print ('Bonjour')
2 print ('Bonjour')
3 print ('Bonjour')
4 print ('Bonjour')
5 print ('Bonjour')

Console
Bonjour
Bonjour
Bonjour
Bonjour
Bonjour

```

```

1 for i in range (5):
2     print ('Bonjour')

Console
Bonjour
Bonjour
Bonjour
Bonjour
Bonjour

```

→ Remarque

```

1 for i in range (5):
2     print ('Bonjour')
3     print ('Salut')

```

```

1 for i in range (5):
2     print ('Bonjour')
3     print ('Salut')

```

Vous souvenez-vous la notion d'indentation que nous avons vu avec if ?

Si vous tapez sur entrée après les deux points, le curseur se déplace vers la droite, tant que nous gardons cet espace, les instructions seront répétées.

4.2. Notion du compteur

La variable i désigne le compteur de la boucle : par défaut i commence du 0 jusqu'à n-1, l'incrémentation du compteur est automatique avec un pas = 1.

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

Console

```
0
1
2
3
4
```

Cette notion du compteur sera très utile pour parcourir une chaîne de caractères (caractère par caractère).

```
1 ch = 'python'
2 print(ch[0])
3 print(ch[1])
4 print(ch[2])
5 print(ch[3])
6 print(ch[4])
7 print(ch[5])
```

Console

```
p
y
t
h
o
n
```

```
1 ch = 'python'
2 for i in range (6):
3     print(ch[i])
```

Console

```
p
y
t
h
o
n
```

Dans le cas d'une chaîne donnée par l'utilisateur qu'on ne sait pas d'avance, on utilise ce code pour parcourir la chaîne.

```
1 ch = input ('Donner une chaîne')
2 for i in range (len(ch)):
3     print(ch[i])
```

Exercice : Somme de 5 entiers

Ecrire un programme qui permet de calculer la somme de 5 entiers donnés.

L'idée est de demander à l'utilisateur de saisir un entier 5 fois, puis on calcule la somme de ces entiers.

Etape n°1 :

```
1 for i in range (5):
2     n = int (input('Donner un entier'))
3     s = n
4     print(s)
```

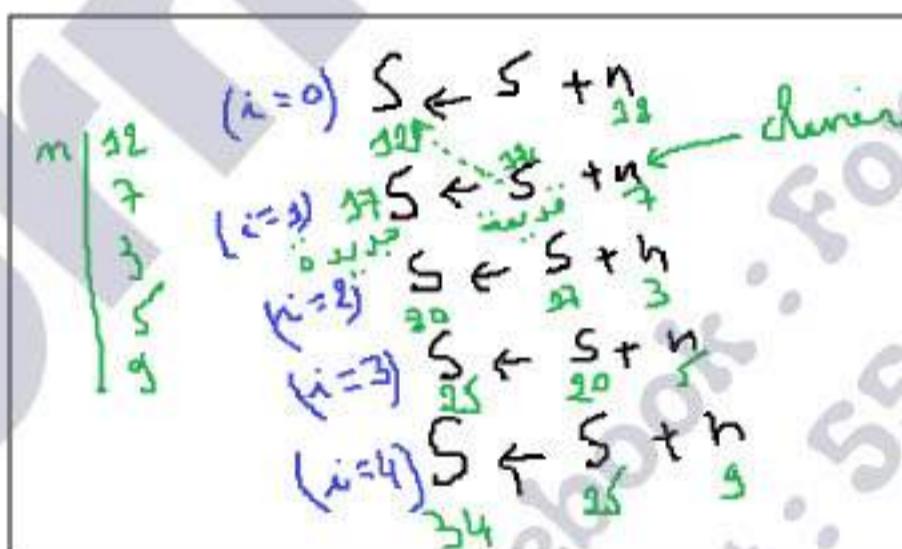
```
for i in range(5):
    n = int(input('—'))
```

$$\begin{array}{l} i=0 \quad | \quad s \\ i=1 \quad | \quad \cancel{s} \oplus \end{array}$$

$$s = n$$

Etape n°2 :

```
1 for i in range (5):
2     n = int (input('Donner un entier'))
3     s = s + n
4     print (s)
```



L'étape de calcul de la somme est répétée 5 fois, à chaque fois on met à jour la valeur de la somme après chaque entier ajouté par l'utilisateur

Etape n°3 :

```
1 for i in range (5):
2     n = int (input('Donner un entier'))
3     s = s + n
4     print(s)
```

```
Code:
      s = s + n
NameError: name 's' is not defined
```

Ce message d'erreur concerne la variable s qu'est à l'intérieur, on effet pour i = 0, s n'est pas encore définie.

Etape n°4 :

```

1 s = 0
2 for i in range (5):
3     n = int (input('Donner un entier'))
4     s = s + n
5     print (s)

Console:
Donner un entier12
12
Donner un entier7
19
Donner un entier3
22
Donner un entier5
27
Donner un entier9
36

```

La variable s doit être initialiser avant la boucle for, sinon chaque itération on remet cette variable à 0.

On remarque que si on écrit l'instruction print() à l'intérieur de la boucle, on affiche la valeur obtenue à chaque fois alors que nous souhaitons d'afficher la somme totale.

Etape n°5 :

```

1 s = 0
2 for i in range (5):
3     n = int (input('Donner un entier'))
4     s = s + n
5 print (s)

Console:
Donner un entier12
Donner un entier7
Donner un entier3
Donner un entier5
Donner un entier9
36

```

Correction de l'exercice

Algorithme :

Début somme

$s \leftarrow 0$

Pour i de 0 à 4 faire

Ecrire('Donner un entier')

Lire(n)

$s \leftarrow s+n$

Fin Pour

Ecrire(s)

Fin somme

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
s	Entier	La somme de 5 entiers
i	Entier	Compteur

Exercice : Somme des chiffres

Ecrire un programme qui permet de calculer la somme des chiffres d'un entier composé de 3 chiffres (sans utiliser mod et div).

Correction de l'exercice

Algorithme :

Début somme_chiffres

$s \leftarrow 0$

Ecrire('Donner un entier')

Lire(n)

ch \leftarrow convch(n)

$s \leftarrow valeur(ch[0])+valeur(ch[1])+valeur(ch[2])$

Ecrire(s)

Fin somme_chiffres

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
ch	chaine de caractères	
s	Entier	La somme des chiffres de n

Python :

```

1 x = int (input ('Donner un entier composé de 3 chiffres'))
2 ch = str (x)
3 s = int (ch[0]) + int(ch[1]) + int(ch[2])
4 print ('La somme des chiffres est',s)
5

```

Console:

Donner un entier composé de 3 chiffres253
La somme des chiffres est 10



Il faut convertir (ch[0]), ch[1] et ch[2] en entier sinon nous sommes entrain de reconstruire ch.

Exercice : Somme des chiffres

Ecrire un programme qui permet de calculer la somme des chiffres d'un entier quelconque

Correction de l'exercice

Algorithme :

Début somme_chiffres

s ← 0

Ecrire('Donner un entier')

Lire(n)

ch \leftarrow convch(n)

Pour i de 0 à long(ch)-1 faire

s \leftarrow s + valeur(ch[i])

Fin Pour

Ecrire(s)

Fin somme_chiffres

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
ch	chaine de caractères	
s	Entier	La somme des chiffres de n
i	Entier	Compteur

Python :

```

1 x = int (input ('Donner un entier composé de 3 chiffres'))
2 ch = str (x)
3 s=0
4 for i in range (len(ch)):
5     s = s + int (ch[i])
6 print ('La somme des chiffres est',s)

```

Exercice : Nombre des espaces

Ecrire un programme qui permet de calculer le nombre des espaces dans une chaîne de caractères donnée.

Exemple : ch=' Salut tout le monde' on aura Le nb d'espace est 3.

Correction de l'exercice

Algorithme :

```

Début nb_espace
nb ← 0
Ecrire('Donner une chaîne de caractère')
Lire(ch)
Pour i de 0 à long(ch)-1 faire
    si ch[i] = ' ' alors nb ← nb+1
Fin Si
Fin Pour
Ecrire ('Le nb d espace est ',nb)
Fin nb_espace

```

TDO

Objet	Type/Nature	Rôle
nb	Entier	le nombre des espaces
ch	chaîne de caractères	Donnée

Python :

```

1 ch = input ('Donner une chaîne de caractère')
2 nb = 0
3 for i in range (len(ch)):
4     if ch[i] == ' ':
5         nb = nb + 1
6 print ('Le nombre d espace est',nb)
7

```

Console

Donner une chaîne de caractère Salut tout le monde
Le nombre d espace est 3

4.3. Modification de la valeur d'une chaîne de caractères

En python, une chaîne de caractères est un objet de la catégorie des séquences (collections ordonnées d'éléments) **non modifiables** (on dit aussi **non mutable**). Contrairement à d'autres langages, en Python, on ne peut donc pas modifier directement un caractère individuel dans une chaîne. Le seul moyen de modifier une variable de ce type sera de la redéfinir (création d'une nouvelle variable de même nom qui écrase l'ancienne ou chaîne copie).

L'exemple suivant illustre la modification de la valeur (complète) d'une chaîne de caractères. Cette modification consiste à remplacer tous les espaces par le caractère '_'. La chaîne à modifier est stockée dans la variable *ch*. Une seconde variable *ch2* est initialisée vide avant l'entrée dans une boucle *for*. Cette boucle parcourt la valeur de *chaine* caractère par caractère, et si le caractère pointé est différent de l'espace il est ajouté par concaténation à la valeur de *ch2* sinon c'est le caractère '_' qui est ajouté à *ch2*. Après exécution de la boucle, la valeur de *ch2* est recopiée dans *ch* qui est ainsi modifiée.

Pour ce faire et mieux comprendre, on va passer par quelques étapes :

Vous ne pouvez pas modifier le contenu d'une chaîne existante. En d'autres termes, vous ne pouvez pas utiliser l'opérateur [] dans la partie gauche d'une instruction d'affectation. Essayez par exemple d'exécuter le petit script suivant (qui cherche intuitivement à remplacer une lettre dans une chaîne) :

```
1 ch = 'python'
2 ch[1] = 'i'
3 print(ch)

Console:
      ch[1] = 'i'
TypeError: 'str' object does not support item assignment
```

Le résultat attendu par le programmeur qui a écrit ces instructions est « pithon » (avec un i au lieu de y). Mais contrairement à ses attentes, ce script lève une erreur du genre : *TypeError: 'str' object does not support item assignment*.

En algorithme, on peut effectuer ce changement sans aucun problème.

ch ← "python"

`ch[1] ← "i"`

ch se transforme en "python"

Exercice : Remplacer les caractères

Ecrire un programme qui permet de remplacer 'y' dans 'python' par 'i'.

Correction de l'exercice

Python :

```

1 ch = 'python'
2 print (ch)
3 ch1 = ch[0:1]
4 print (ch1)
5 ch2 = 'i'
6 ch3 = ch[2:len(ch)]
7 print (ch3)
8 ch4 = ch1 + ch2 + ch3
9 print (ch4)

```

Console
python
P
thon
python

Exercice : Concaténation

Ecrire un programme qui permet de concaténer n chaînes de caractères pour former une chaîne de caractères.

Le résultat obtenu par cet exercice est une nouvelle chaîne différente de celle d'origine.

Correction de l'exercice

Algorithme :

Début concaténation

Ecrire ('Donner le nbr des chaines')

Lire (n)

`ch ← "`

Pour i de 0 à n-1 faire

Ecrire ('Donner une chaîne')

Lire (c)

ch1 ← ch1 + c

Fin Pour

Ecrire (ch)

Fin concaténation

TDO

Objet	Type/Nature	Rôle
n	Entier	le nombre des chaînes
ch	chaîne de caractères	la somme des chaînes
c	chaîne de caractères	Donnée

Python :

```

1 n=int(input('Donner le nombre de chaînes'))
2 for i in range(n):
3     print('Donner la chaîne numéro ',i+1)
4     c=input()
5     ch1=ch1+c
6 print(ch1)

C:\Users\asus\Desktop\Formakt Bac>concatenation_chaine.py
Donner le nombre de chaînes?
Donner la chaîne numéro 1
Python
Traceback (most recent call last):
  File "C:\Users\asus\Desktop\Formakt Bac\concatenation_chaine.py", line 5, in <module>
    ch1=ch1+c
NameError: name 'ch1' is not defined

```



Comme nous avons vu pour l'exercice du calcul de la somme des 5 entiers, un message d'erreur s'affiche. En effet pour la première itération ($i=0$) la nouvelle chaîne ch1 n'est pas encore définie. Afin de résoudre ce problème nous devons initialiser ch1 à une valeur qui n'affecte pas le résultat souhaité (La valeur 0 pour les entiers dans le calcul d'une somme)

Ch1 sera initialisé à une chaîne vide.

```

1 n = int (input ('Donner le nombre de chaîne'))
2 ch=''
3 for i in range (n):
4     print ('Donner une chaîne numéro', i+1)
5     c = input()
6     ch = ch + c
7 print (ch)

```

4.3.1. Autre méthode de la boucle for

Copie chaîne.py >

```

1 ch=input('Donner une chaîne de caractères')
2 ch1= ''
3 for carac in ch:
4     ch1= ch1+ carac
5 print(ch1)

```

Former une chaîne (copie) de la chaîne saisie

Et maintenant on fait l'exemple du début de la partie modification d'une chaîne :

```

ch=input('Donner une chaîne de caractère')
ch1=''
for carac in ch :
    if carac==' ':
        carac='_'
        ch1=ch1+carac
print(ch1)

```

➤ Changement des espaces dans une chaîne par des "_"

Autre méthode :

Nous pouvons modifier les espaces dans ch par une autre méthode dont l'algorithme est le suivant :

Ecrire ("Donner une chaîne")

```

lire (ch)
ch ← ""

Pour i de 0 à long(ch) – 1 faire

    Si ch[i] = " " alors

        ch2 ← ch2 + " _"

    Sinon

        ch2 ← ch2 + ch[i]

    Fin Si

Fin Pour

```

Exercice : Modification chaîne de caractères

Ecrire un programme qui permet de changer le caractère de la position donnée par ‘_’.

Exemple : ch='python' et la position est 3, on aura py_hon

Correction de l'exercice

Algorithme :

```

Début modification

Ecrire ('Donner une chaîne de caractères')
Lire (ch)
Ecrire (Donner la position')
Lire(pos)
ch2 ← ""

Pour i de 0 à long(ch)-1 faire

    si ch[i] = ch[pos-1] alors

        ch2 ← ch2 + " _"

    Sinon

        ch2 ← ch2 + ch[i]

```

Fin Si
Fin Pour
Fin modification

Python

```

1 ch=input('Donner une chaîne de caractères')
2 pos=int(input('Donner la position '))
3 ch2=''
4 for i in range(len(ch)):
5     if ch[i]==ch[pos-1]:
6         ch2=ch2+'_'
7     else:
8         ch2=ch2+ch[i]
9 print(ch2)

```

Il a à noter qu'avec cette méthode le programme remplace tous les caractères identiques au caractère trouvé dans la position pos.

```

1 ch=input('Donner une chaîne de caractères')
2 pos=int(input('Donner la position '))
3 ch2=''
4 for i in range(len(ch)):
5     if ch[i]==ch[pos-1]:
6         ch2=ch2+'_'
7     else:
8         ch2=ch2+ch[i]
9 print(ch2)

```

Console <
 >>> %Run modif.py
 Donner une chaîne de caractèresBonjour
 Donner la position 2
 B_nj_ur

Ce résultat a obtenu le fait que nous avons comparé `ch[i]` par `ch[pos-1]` qu'est un caractère. Pour répondre au besoin de cet exercice, il faut qu'on remplace juste le caractère trouvé dans la position pos. Pour cela on doit utiliser `pos-1` et non pas `ch[pos-1]`.

Algorithme :

Début modification

Ecrire ('Donner une chaîne de caractères')

Lire (ch)

Ecrire ('Donner la position')

Lire(pos)

ch2 \leftarrow ""

Pour i de 0 à long(ch)-1 faire

 si i = pos-1 alors

 ch2 \leftarrow ch2 + "_"

 sinon

 ch2 \leftarrow ch2+ch[i]

 Fin Si

Fin Pour

Fin modification

TDO

Objet	Type/Nature	Rôle
ch	Chaine de caractères	Donnée
pos	Entier	Donnée
Ch2	Chaine de caractères	La chaîne modifiée

python :

```

1 ch=input('Donner une chaîne de caractères')
2 pos=int(input('Donner la position '))
3 ch2=''
4 for i in range(len(ch)):
5     if i==pos-1:
6         ch2=ch2+'_'
7     else:
8         ch2=ch2+ch[i]
9 print(ch2)

Console - com
>>> %Run modif.py
Donner une chaîne de caractèresbonjour
Donner la position 2
b_njour

```

Exercice : Modification chaîne de caractères

Ecrire un programme qui permet de changer le caractère espace par son nombre d'occurrence.

Exemple : ch = 'Salut tout le monde', on aura salut1tout2le3monde

Correction de l'exercice

Algorithme :

Début chang_occ

Ecrire ('Donner une chaîne de caractères')

Lire (ch)

nb \leftarrow 0

ch2 \leftarrow ''

Pour i de 0 à long(ch)-1 faire

 si ch[i]= ' ' alors

 nb \leftarrow nb+1

 ch2 \leftarrow ch2 + convch(nb)

 sinon

 ch2 \leftarrow ch2+ch[i]

```

Fin Si
Fin Pour
Ecrire (ch)
Fin chang_occ

```

TDO

Objet	Type/Nature	Rôle
ch	Chaine de caractères	Donnée
nb	Entier	nombre d'occurrence d'un espace
ch2	Chaine de caractères	La chaine modifiée

Python :v

```

1 ch=input('Donner une chaine de caractère ')
2 ch2=''
3 nb=0
4 for i in range(len(ch)):
5     if ch[i]==' ':
6         nb=nb+1
7         ch2=ch2+str(nb)
8     else:
9         ch2=ch2+ch[i]
10 print(ch2)

Console
>>> %Run occu.py
Donner une chaine de caractère salut tout le monde
saluttout2le3monde

```

Exercice d'application

Ecrire un programme Python qui permet de transformer la chaîne de caractères donnée en majuscule sans utiliser la fonction `c.upper()` ou `ch.upper()`.

Indication : le code ASCII de 'a' est 97 et le code ASCII de 'A' est 65.

- Transformer une chaîne en majuscule

```

1 ch=input('Donner une chaîne de caractères')
2 ch1=''
3 for i in range(len(ch)):
4     if 'A'<=ch[i]<='Z':
5         ch1=ch1+ch[i]
6     else:
7         ASCIIImin=ord(ch[i])
8         ASCIIImajus =(ASCIIImin -32)
9         jdid = chr(ASCIIImajus)
10        ch1=ch1+jdid
11 print(ch1)

```

Console
Donner une chaîne de caractèrespython
PYTHON

➤ Changer les caractères d'une chaîne donnée par leurs codes ASCII

```

1 ch=input('Donner une chaîne de caractères')
2 ch1=''
3 for i in range(len(ch)):
4     ch1=ch1+str(ord(ch[i]))
5 print(ch1)

```

Console
Donner une chaîne de caractèrespython
112121116104111110

Exercice : Factorielle

Ecrire un programme qui permet de calculer la factorielle d'un entier donnée

$$6! = 1 * 2 * 3 * 4 * 5 * 6$$

Correction de l'exercice

Algorithme :

Début factorielle

Ecrire ('Donner un entier')

Lire (n)

$f \leftarrow 1$

Pour i de 0 à n-1 faire

$f \leftarrow f * (i+1)$

Fin Pour

Fin factorielle

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
f	Entier	Factorielle de n

Python :

```

1 n = int(input('Donner un entier'))
2 f = 1
3 for i in range (n):
4     f = f * (i+1)
5 print (f)

```

```

Cela donne :
Donner un entier6
720

```

Méthode 2

Remarques

Le compteur de la boucle Pour commence par défaut par 0, mais on peut également imposer la valeur de départ du compteur.

```

1 n = int(input('Donner un entier'))
2 f = 1
3 for i in range (1,n+1):
4     f = f * i
5 print (f)

```

for compteur in range (début, fin , p)

#Méthode 1

1 for i in range (2):

2 print (i)

NB : compteur variant entre [début..fin-1]

et pas = p

le programme affiche :

0

1

#Méthode 2

1 for i in range (1,10,2) :

2 print(i)

le programme affiche :

1

3

5

7

9

#Méthode 3

1 for i in range (1,5):

2 print(i)

le programme affiche :

1

2

3

4

$$6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6$$

$$6! = 1 \times \cancel{2} \times 2 \times 3 \times 4 \times 5 \times 6$$

$f = f * i \quad (i=2)$

زائد

```

1 n = int(input('Donner un entier'))
2 f = 1
3 for i in range (2,n+1):
4     f = f * i
5 print (f)

```

Méthode 3 :

$$6! = 1 \times 2 \times 3 \times 4 \times 5 \times 6$$

$$(n - [n-1]) \cdot [\dots] \cdot (n-2) \times (n-1) \times (n-0)$$

$$(n - i)$$

```

1 n = int(input('Donner un entier'))
2 f = 1
3 for i in range (0,n):
4     f = f * (n-i)
5 print (f)

```

Exercice : Diviseurs

Ecrire un programme qui permet de vérifier si un entier est diviseur d'un autre.

Correction de l'exercice

Algorithme :

Début diviseur

Ecrire ('Donner un entier')

Lire (x)

Ecrire ('Donner un entier')

Lire(y)

Si $x \bmod y = 0$ alors Ecrire(x, 'est divisible par', y)

Fin Si

Fin diviseur

TDO

Objet	Type/Nature	Rôle
x	Entier	Donné
y	Entier	Donné

Python :

```

1 x = int(input('Donner un entier'))
2 y = int(input('Donner un autre'))
3 if x % y == 0 :
4     print(y, 'est diviseur de', x)

```

Exercice : Afficher les diviseurs

Ecrire un programme qui permet d'afficher les diviseurs d'un entier donné.

Correction de l'exerciceAlgorithme :

Début diviseur

Ecrire ('Donner un entier')

Lire (x)

Pour i de 1 à x faire

 si x mod i = 0 alors Ecrire (i)

 Fin Si

Fin Pour

Fin diviseur

TDO

Objet	Type/Nature	Rôle
x	Entier	Donné
i	Entier	Compteur

Python :

```

1 x = int (input('Donner un entier'))
2 for i in range (1,x+1):
3     if x % i == 0 :
4         print (i)

```

Console

```

Donner un entier6
1
2
3
6

```

Exercice : Somme des diviseurs

Ecrire un programme qui permet de calculer la somme des diviseurs d'un entier donné.

Correction de l'exercice

Algorithme :

Début diviseur

Ecrire ('Donner un entier')

Lire (x)

s \leftarrow 0

Pour i de 1 à x faire

 si x mod i = 0 alors s \leftarrow s+i

 Fin Si

Fin Pour

Ecrire('La somme des diviseurs de' ,x, 'est' , s)

Fin diviseur

TDO

Objet	Type/Nature	Rôle
x	Entier	Donné
s	Entier	La somme des diviseurs de x
i	Entier	Compteur

Python :

```

1 x = int(input('Donner un entier'))
2 s = 0
3 for i in range(1,x+1):
4     if x % i == 0 :
5         s = s + i
6 print ('La somme des diviseurs de ',x,'est',s)

```

Console

Donner un entier6
La somme des diviseurs de 6 est 12

Exercice : Nombre des diviseurs

Ecrire un programme qui permet de calculer le nombre des diviseurs d'un entier donné.

Correction de l'exerciceAlgorithme :

Début diviseur

Ecrire ('Donner un entier')

Lire (x)

nb ← 0

Pour i de 1 à x faire

 si $x \bmod i = 0$ alors nb ← nb+1

 Fin Si

Fin Pour

Ecrire('Le nombre des diviseurs de ',x,'est',nb)

Fin diviseur

TDO

Objet	Type/Nature	Rôle
x	Entier	Donné
nb	Entier	Le nombre des diviseurs de x
i	Entier	Compteur

Python :

```

1 x = int(input('Donner un entier'))
2 nb = 0
3 for i in range(1,x+1):
4     if x % i == 0 :
5         nb = nb + 1
6 print ('Le nombre des diviseurs de',x,'est',nb)

```

Console

```

Donner un entier6
Le nombre des diviseurs de 6 est 4

```

Exercice Nombre Premier

Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs. Ces deux diviseurs sont 1 et le nombre considéré

Correction de l'exerciceAlgorithme :

Début premier

Ecrire ('Donner un entier')

Lire (n)

nb \leftarrow 0

Pour i de 0 à n-1 faire

 si n mod (i+1) = 0 alors nb \leftarrow nb+1

 Fin Si

Fin Pour

Ecrire('Le nombre des diviseurs de' ,x,' est' ,nb)

Si nb = 2 alors Ecrire(n, 'est premier')

Sinon Ecrire (n, 'n est pas premier')

Fin Si

Fin premier

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné

nb	Entier	Le nombre des diviseurs de ,
i	Entier	Compteur

Python :

```

1 n = int(input('Donner un entier à vérifier'))
2 #Chercher le nombre des diviseurs de l'entier
3 nbd = 0
4 for i in range (n):
5     if n % (i+1) == 0:
6         nbd = nbd + 1
7 print ('Le nombre des diviseurs de', n,'est',nbd)
8 #Tester si l'etier est premier ou non
9 if nbd == 2:
10    print (n , 'est premier')
11 else:
12    print (n , 'est n est pas premier')

```

Méthode 2 :Algorithme :

Début premier

Ecrire ('Donner un entier')

Lire (n)

nb \leftarrow 0

Pour i de 2 à n-1 faire

 si n mod i = 0 alors nb \leftarrow nb+1

Fin Si

Fin Pour

Ecrire('Le nombre des diviseurs de ' ,x, ' est' , nb)

Si nb = 0 alors Ecrire(n, 'est premier')

Sinon Ecrire (n,'n est pas premier')

Fin Si

Fin premier

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné

nb	Entier	Le nombre des diviseurs de ,
i	Entier	Compteur

Python :

```

1 n = int (input('Donner un entier à vérifier'))
2 #Chercher le nombre des diviseurs de l'entier
3 nbd = 0
4 for i in range (2,n):
5     if n % i == 0:
6         nbd = nbd + 1
7 #Tester si l'etier est premier ou non
8 if nbd == 0:
9     print (n , 'est premier')
10 else:
11     print (n , 'n est pas premier')
```

4.4. Boucle Tant que

La boucle **while** répète le bloc d'instruction lorsque la condition d'entrée est vérifiée et retourne après une itération pour vérifier encore. L'itération s'arrête lorsque la condition n'est plus vérifiée

Algorithme	Python
Tant que <i>condition d'entrée faire</i> <i>Traitements à répéter</i> Fin Tant que	while <i>condition d'entrée :</i> <i>Traitements à répéter</i>

Pour comprendre le fonctionnement de while :

L'objectif ici est d'atteindre i égale à 5

Boucle infinie

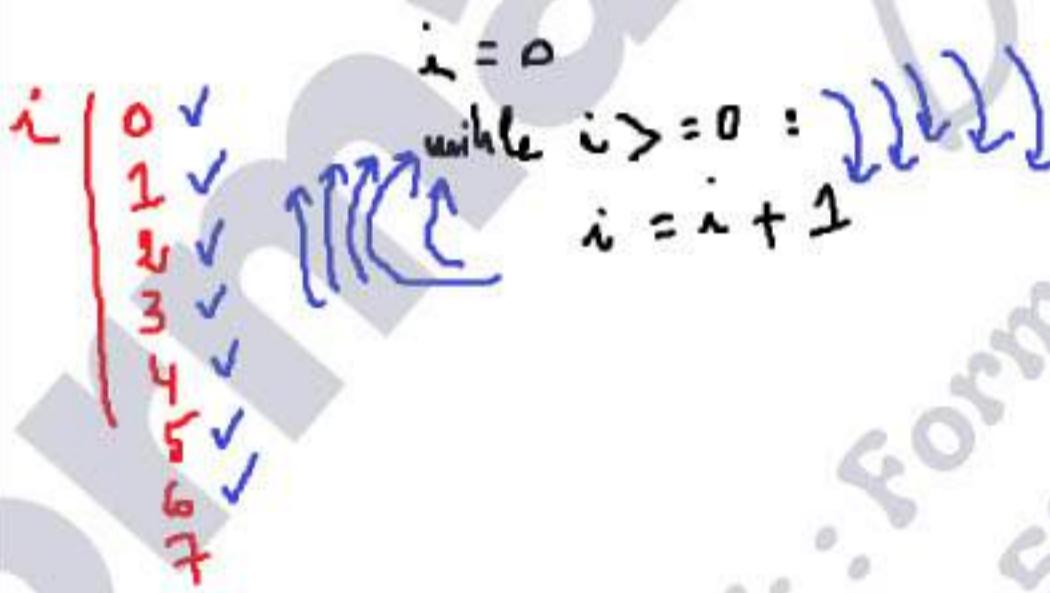
```

1 i = 0
2 while i >= 0:
3     i = i + 1
4     print (i)

```

Console
533
534
535
536
537
538
539
540
541
542

Tant que la condition d'entrée est vraie la boucle ne s'arrête pas.



```

1 i = 0
2 while i <= 5:
3     i = i + 1
4     print (i)

```

Console
1
2
3
4
5
6

```

1 i = 0
2 while i != 5:
3     i = i + 1
4     print (i)

```

Console
1
2
3
4
5

4.5. Boucle répéter

Sans faire une vérification pour la première itération, la boucle répéter refait le bloc d'instruction jusqu'à la condition de sortie sera vérifiée, dans ce cas on n'aura plus de répétition

En algorithme, la boucle répéter s'écrit :

Répéter

Traitements à répéter

Jusqu'à *condition de sortie*

Le traitement se répète si la condition de sortie n'est pas vérifiée et il s'arrête au cas contraire.

En pratique, python n'a pas une instruction pré définie qui remplace directement la boucle Répéter (le langage de programmation Pascal par exemple, a une instruction directe de la boucle répéter, c'est repeat *Traitements until condition*)

On doit construire un bloc d'instruction qui fait le même rôle de la boucle Répéter en utilisant la boucle while :

Méthode n°1 :

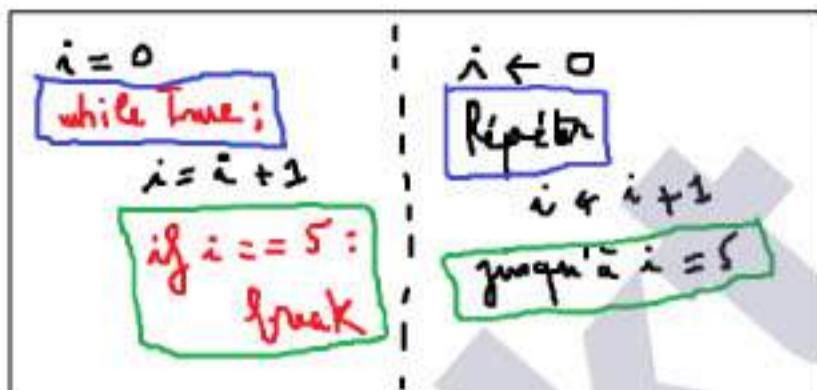
La boucle répéter ne nécessite pas une condition d'entrée autrement dit elle est toujours vraie, pour cela nous procérons comme suit :

```
1 while True:  
2     traitement  
3
```

La boucle répéter s'arrête lorsqu'une condition de sortie est vérifiée, pour cette méthode l'instruction jusqu'à est remplacée par une combinaison de deux instructions : if et break

```
while True:  
    traitement  
    if condition de sortie:  
        break
```

Break : C'est une instruction permet de sortir de toute itération (*for, while...*) qui la contient. Son effet est limité à un seul niveau d'imbrication.



Lorsque la variable *i* atteint la valeur 5, la boucle répéter s'arrête.

Méthode n°2 :

Devant le while, on met une condition qui est vérifiée pour qu'on peut entrer à la boucle directement, c'est comme la boucle répéter qui n'a pas en théorie une condition d'entrée

Donc on affecte à une variable booléenne True et on compare test avec True dans la condition d'entrée de while.

```

1 test = True
2 while test == True :
3
  
```

On a créé comme ça l'instruction Répéter

Maintenant, on doit construire une instruction qui remplace le rôle de Jusqu'à :

Le principe ici c'est d'affecter à la variable test une autre valeur (test = False) lorsque la condition de sortie est vérifiée, donc dans l'itération suivante, test sera false et on ne peut entrer dans la boucle car on a : while test == True, qui n'est pas maintenant vérifiée donc la boucle s'arrête.

```

1 test = True
2 while test == True :
3   Traitement
4   if condition de sortie :
5     test = False
  
```

Comme ça on construit une boucle Répéter avec ce principe.

La même chose si on affecte à la variable test la valeur False au début et on la change avec True si la condition de sortie est vérifiée :

```

1 test = False
2 while test == False :
3     Traitement
4     if condition de sortie :
5         test = True

```

C'est aussi la boucle Répéter, le même principe

Autre écriture équivalente :

En programmation,

if test == True :
Résultat

est équivalente à :

if test :
Résultat

Aussi,

if test == False :
Résultat

est équivalente à :

if not (test) :
Résultat

Donc on peut écrire en pratique la boucle Répéter par les 2 façons équivalentes suivantes :

```

1 test = True
2 while test :
3     Traitement
4     if condition de sortie :
5         test = False

```

Ou

```

1 test = False
2 while not (test) :
3     Traitement
4     if condition de sortie :
5         test = True

```

Algorithme	Python
Répéter <i>Traitement à répéter</i> Jusqu'à condition de sortie	test = True while test : <i>Traitement à répéter</i> if condition de sortie : test = False

Exemple en Algorithme

i ← 0
Répéter
i ← *i* + 1
Jusqu'à *i* = 5

Le même exemple en Python :

```

1 test=True
2 i=0
3 while test:
4     i=i+1
5     if i==5:
6         test=False
7 print(i)

```

Console
 Python 3.7.9 (bundled)
 >>> %Run Cours.py

5

Saisie conditionnée ou contrôlé:

Algorithme	Python
Ecrire ("Donner votre note")	1 moy = float(input ('donner votre note'))
Lire (moy)	2 while (moy <0 or moy>20):
Tant que (moy<0) ou (moy>20) faire	3 print('Erreur, donner votre note')
Ecrire ("Donner votre note")	
Fin tant que	
Répéter	1 test=True
Ecrire ("Donner votre note")	2 while test :
Lire (moy)	3 moy=float(input('donner votre note'))
Jusqu'à (moy>=0) et (moy<=20)	4 if moy >=0 and moy<=20 :
	5 test=False

La condition d'entrée pour la boucle tant que est opposée à la condition de sortie de la boucle répéter.

Exercice PGCD :

Ecrire un programme qui permet de calculer le pgcd de 2 entiers a et b par la méthode de différence.

Exemple : a=15 et b=27

$\text{pgcd}(15,27)=\text{pgcd}(15,27-15)=\text{pgcd}(15,12)=\text{pgcd}(15-12,12)=\text{pgcd}(3,12)=\text{pgcd}(3,12-3)=\text{pgcd}(3,9)=\text{pgcd}(3,9-3)=\text{pgcd}(3,6)=\text{pgcd}(3,6-3)=\text{pgcd}(3,3)=3$

Si $a > b$ alors $a \leftarrow a - b$
 sinon $b \leftarrow b - a$
 fin si

Répéter

si $a > b$ alors $a \leftarrow a - b$

sinon $b \leftarrow b - a$

fin si

jusqu'à $a = b$

Correction de l'exercice avec la boucle répéter

Algorithme :

Début PGCD

Ecrire ('Donner a')

Lire (a)

Ecrire('Donner b')

Lire(b)

Répéter

si $a > b$ alors $a \leftarrow a - b$

sinon $b \leftarrow b - a$

jusqu'à $a = b$

Ecrire(a)

Fin PGCD

TDO

Objet	Type/Nature	Rôle
a	Entier	Donné
b	Entier	Donné

Python :

```

1 a=int(input('Donner a'))
2 b=int(input('Donner b'))
3 test=True
4 while test:
5     if a>b:
6         a=a-b
7     elif b>a:
8         b=b-a
9     if a==b:
10        test=False
11 print(a)

```

Correction de l'exercice avec la boucle tant queAlgorithme :

Début PGCD

Ecrire ('Donner a')

Lire (a)

Ecrire('Donner b')

Lire(b)

Tant que a≠b faire

si a>b alors a ← a-b

sinon b ← b-a

Fin tant que

Ecrire('le pgcd ',a)

Fin PGCD

TDO

Objet	Type/Nature	Rôle
a	Entier	Donné
b	Entier	Donné

Python :

```

1 a = int (input('Donner a '))
2 b = int (input('Donner b '))
3 while a != b:
4     if a>b:
5         a = a-b
6     else:
7         b = b-a
8 print('Le PGCD est',a)

```

Console

Donner a 15
Donner b 27
Le PGCD est 3

Exercice nombre parfait :

Ecrire un programme qui permet d'afficher tous les nombres parfaits entre m et n tel que $2 \leq m < n$.

On dit qu'un nombre est parfait s'il est égal à la somme de tous ses diviseurs autre que lui-même.

Etape n°1 :

Pour tester si un entier n donné est parfait ou non .On procède comme suit :

```

1 p = int (input('Donner un entier'))
2 s = 0
3 for i in range (1,p):
4     if p % i == 0 :
5         s = s + i
6 if s == p:
7     print(p,'est parfait')

```



Les diviseurs d'un entier quelconque autre que lui-même sont inférieurs à sa moitié, donc pour optimiser le nombre d'itération.

```

1 p = int(input('Donner un entier'))
2 s = 0
3 for i in range(1, p // 2 + 1):
4     if p % i == 0:
5         s = s + i
6 if s == p:
7     print(p, 'est parfait')

```

Ceci n'est qu'une partie de l'exercice, car l'exercice nous a demandé d'afficher tous les nombres parfaits entre deux entiers donnés m et n.

Correction de l'exercice

Algorithme :

Début parfait

Répéter

Ecrire('Donner m')

Lire(m)

Ecrire('Donner n')

Lire(n)

jusqu'à $2 \leq m \leq n$

Pour i de m à n faire

s ← 0

Pour j de 1 à i-1 faire

 si $i \bmod j = 0$ alors $s \leftarrow s + j$

 Fin Si

 Fin Pour

 Si $s = i$ alors Ecrire(i)

 Fin Si

 Fin Pour

Fin p&arfait

TDO

Objet	Type/Nature	Rôle

m	Entier	Donné
n	Entier	Donné
s	Entier	La somme des diviseurs
i	Entier	Compteur
j	Entier	Compteur

Python :

```

1 test=True
2 while test:
3     m=int(input('Donner m'))
4     n=int(input('Donner n'))
5     if 2<m<n:
6         test=False
7     for i in range(m,n+1):
8         s=0
9         for j in range(1,i):
10            if i%j==0:
11                s=s+j
12        if s==i:
13            print(i)

```

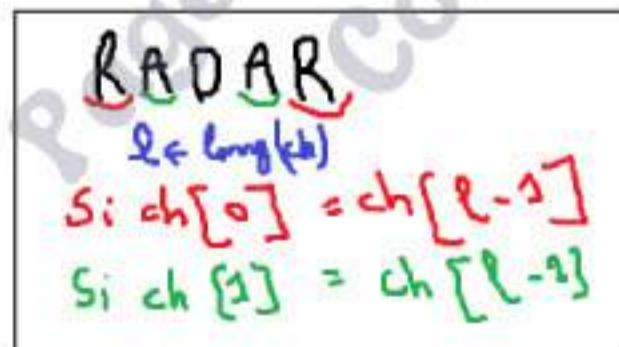
Exercice Palindrome :

Ecrire un programme qui permet de saisir un mot composé de 10 caractères au maximum puis vérifié si ce mot est palindrome (Se lit dans les deux sens)

Exemples : ELLE, RADAR, REVER, DVD, ...

Méthode 1 :

L'idée est de tester les caractères deux à deux comme montre la figure suivante :



Dans le cas d'un mot de taille quelconque

RADAR

$l \leftarrow \text{long}(ch)$
 Si $ch[i] = ch[l-i]$
 Si $ch[i] = ch[l-i]$
 ...
 Si $ch[i] = ch[l-i-1]$

RADAR

$l \leftarrow \text{long}(ch)$
 Si $ch[i] = ch[l-i]$
 Si $ch[i] = ch[l-i]$
 ...
 Si $ch[i] = ch[?]$

Nous allons donc immédiatement envisager d'utiliser la boucle Pour

RADAR

$l \leftarrow \text{long}(ch)$
 Pour i de 0 à $l-1$ faire
 Si $ch[i] = ch[l-i-1]$
 alors écrire (ch , "est palindrome")
 Sinon écrire (ch , "est non palindrome")
 finsi



Ce n'est pas la peine d'aller jusqu'à le dernier caractère de ch puisque la chaîne est symétrique. (Quel que soit la longueur de ch pair ou impair il suffit d'aller juste au milieu de ch)

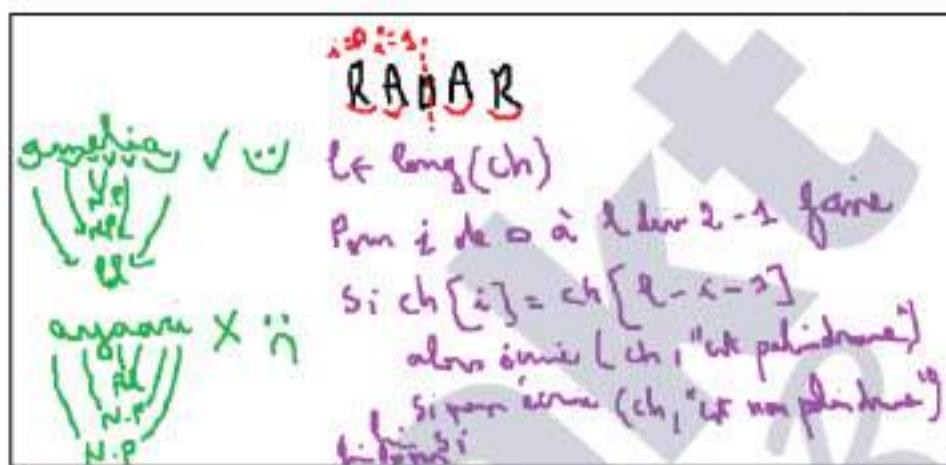
5 div 2 - 1 $\frac{5-1}{2}=2$
 ELLE R A D A R

$l \leftarrow \text{long}(ch)$
 Pour i de 0 à $l \text{ div } 2 - 1$ faire
 Si $ch[i] = ch[l-i-1]$
 alors écrire (ch , "est palindrome")
 Sinon écrire (ch , "est non palindrome")
 finsi

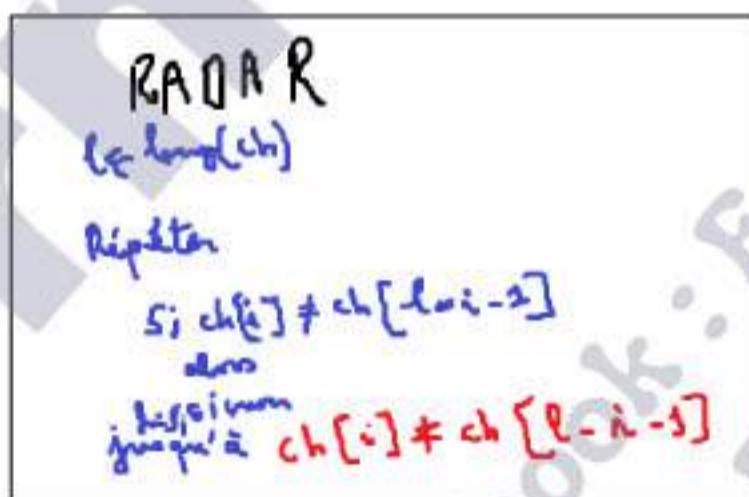
On applique ce code sur deux exemples :

$ch1='amelia'$ et $ch2='ayaari'$.

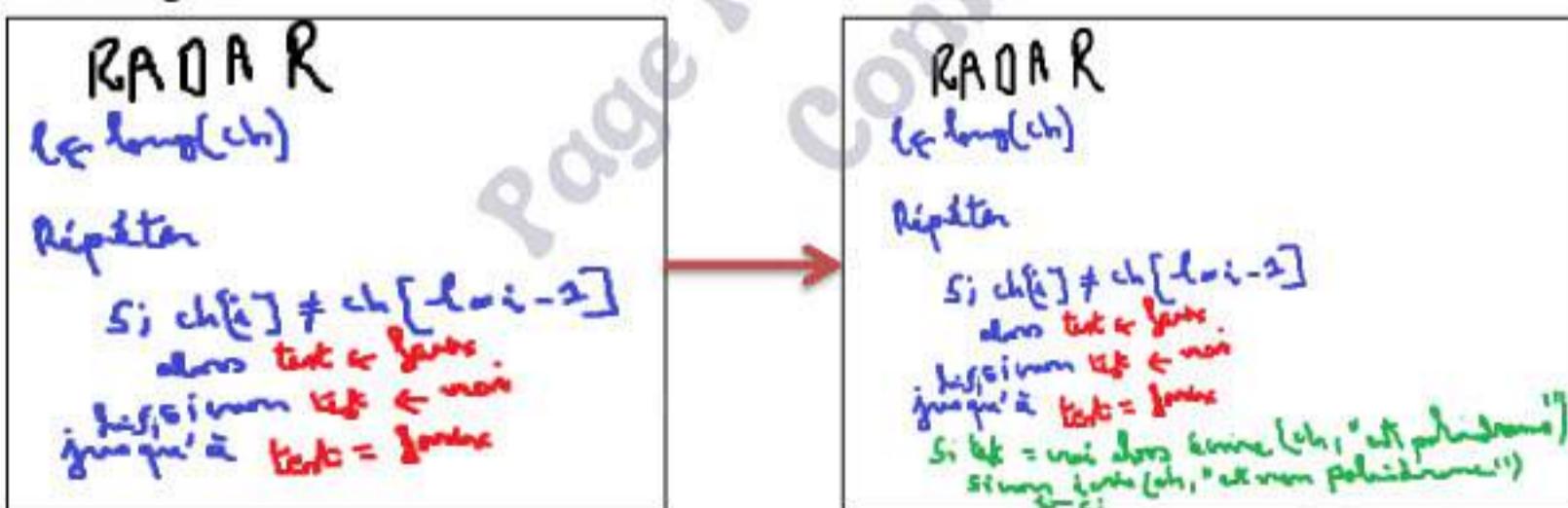
Dans le cas de ch1 le programme affiche Non palindrome ce qu'est vrai alors qu'il affiche Palindrome pour ch2.



La boucle Pour est inefficace dans ce cas, car elle prend en compte le résultat de la vérification des deux derniers caractères (dans le cas de 'ayaari'). On a besoin d'une boucle « de haute dignité », une fois qu'il rencontre deux caractères différents, on sort de la boucle



Le code présenté dans la figure ci-dessus n'affiche pas si ch est palindrome ou non. Pour cela nous introduisons une variable test de type booléen qui servira par la suite pour l'affichage du message.



Attend ! Avant de passer à exécuter ce code la variable i n'est pas encore défini !!!!!!

```

RADAR
l<-long(ch)
i<-2
Répéter
    l<-i+1
    Si ch[i] ≠ ch[-i-2]
        Alors texte <!=> texte
        Si si
            left ← vrai
            jusqu'à (left = vrai) ou (i = l - 1)
            Si vrai
                Si left = vrai alors
                    Si long(ch, "texte <!=> texte")
                    Si non
                        Lire(ch, "texte <!=> texte")
    Si left = vrai alors
        Si long(ch, "texte <!=> texte")
        Si non
            Lire(ch, "texte <!=> texte")

```

Ce genre des boucles nécessite la définition (Initialisation) et l'accrémentation manuel d'un compteur.

Imaginons que nous avons saisir un mot palindrome ? Quand la boucle répéter s'arrête ?

Bah oui, dans ce cas la boucle est infinie et une erreur de type Index out of range nous dérangera.

Donc, il faut prendre en considération ce cas et arrêter la boucle.

```

RADAR
l<-long(ch)
i<-2
Répéter
    l<-i+1
    Si ch[i] ≠ ch[-i-2]
        Alors texte <!=> texte
        Si si
            left ← vrai
            jusqu'à (left = vrai) ou (i = l - 1)
            Si vrai
                Si left = vrai alors
                    Si long(ch, "texte <!=> texte")
                    Si non
                        Lire(ch, "texte <!=> texte")
    Si left = vrai alors
        Si long(ch, "texte <!=> texte")
        Si non
            Lire(ch, "texte <!=> texte")

```

Correction de l'exercice avec la boucle tant que

Algorithme :

Début palindrome

Répéter

Écrire ("Donner un mot "), lire(mot)

Jusqu'a (long(mot) dans [2..10])

1 ← long(mot)

i ← -1

Répéter

i ← i+1

si mot[i] ≠ mot [l-i-1] alors test ← faux

sinon test ← vrai

Finsi

jusqu'à (i=(L div 2)-1) ou (test=faux)

Si (test=vrai) alors écrire("palindrome ")

sinon Écrire(" Non palindrome ")

FinSi

Fin Palindrome

TDO

Objet	Type/Nature	Rôle
mot	Chaine de caractère	Chaîne donnée
l	entier	Longueur de la chaîne
i	entier	compteur
test	booléen	vérification

Python :

```

2  while t1:
3      mot=input('Donner un mot')
4      if 1<len(mot)<=10:
5          t1=False
6  l=len(mot)
7  i=-1
8  t2=True
9  while t2:
10     i=i+1
11     if mot[i]!=mot[l-i-1]:
12         test=False
13     else:
14         test=True
15     if test==False or i==(l//2)-1:
16         t2=False
17 if test==True:
18     print(mot,'est palindrome')
19 else:
20     print(mot,' n est pas palindrome')
21

```

Méthode 2 : Chaine inverse

Algorithme :

Début palindrome

Ecrire('Donner la chaîne à vérifier')

Lire(ch)

ch_inv ← ''

Pour i de long(ch)-1 à 0 (Pas=-1) faire

 ch_inv ← ch_inv + ch[i]

Fin Pour

Ecrire(ch_inv)

Si ch= ch_inv alors Ecrire (ch, ' est palindrome')

Sinon Ecrire(ch, ' n est pas palindrome')

Fin Si

Fin palindrome

TDO

Objet	Type/Nature	Rôle
ch	Chaine de caractère	Donné

ch)inv	chaine de caractères	l'inverse de ch
i	Entier	Compteur

Python :

```

1 ch = input ('Donner une chaine à vérifier')
2 ch_inv = ''
3 for i in range (len(ch)-1, -1 , -1 ) :
4     ch_inv = ch_inv + ch[i]
5 print (ch_inv)
6 if ch == ch_inv :
7     print (ch,'est palindrome')
8 else :
9     print (ch,'est non palindrome')

```

Exercice Etoiles :

Ecrire un programme qui permet d'afficher chacune des 2 figures suivantes pour un nombre de ligne n donné.

Exemple pour n=5

```

*
**
***
****
*****
```

```

*
*** ****
*****
```

Figure 1 :

#Méthode 1 :

Algorithme :

Début figure1

Ecrire('Donner le nombre des lignes')

Lire(n)

Pour i de 1 à n faire

Ecrire('*' * i)

Fin Pour

Fin figure1

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
i	Entier	Compteur

Python :

```
1 n = int(input('Donner le nombre de ligne'))
2 for i in range(1,n+1):
3     print('*' * i)
```

Console

```
Donner le nombre de ligne5
*
**
***
****
*****
*****
```

#Méthode 2 :

Algorithme :

Début figure1

Ecrire('Donner le nombre des lignes')

Lire(n)

ch ← ''

Pour i de 1 à n faire

ch ← ch + '*'

Ecrire (ch)
Fin Pour
Fin figure1

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
ch	Chaine de caractères	chaine contienne les étoiles à afficher pour chaque ligne
i	Entier	Compteur

Python :

```

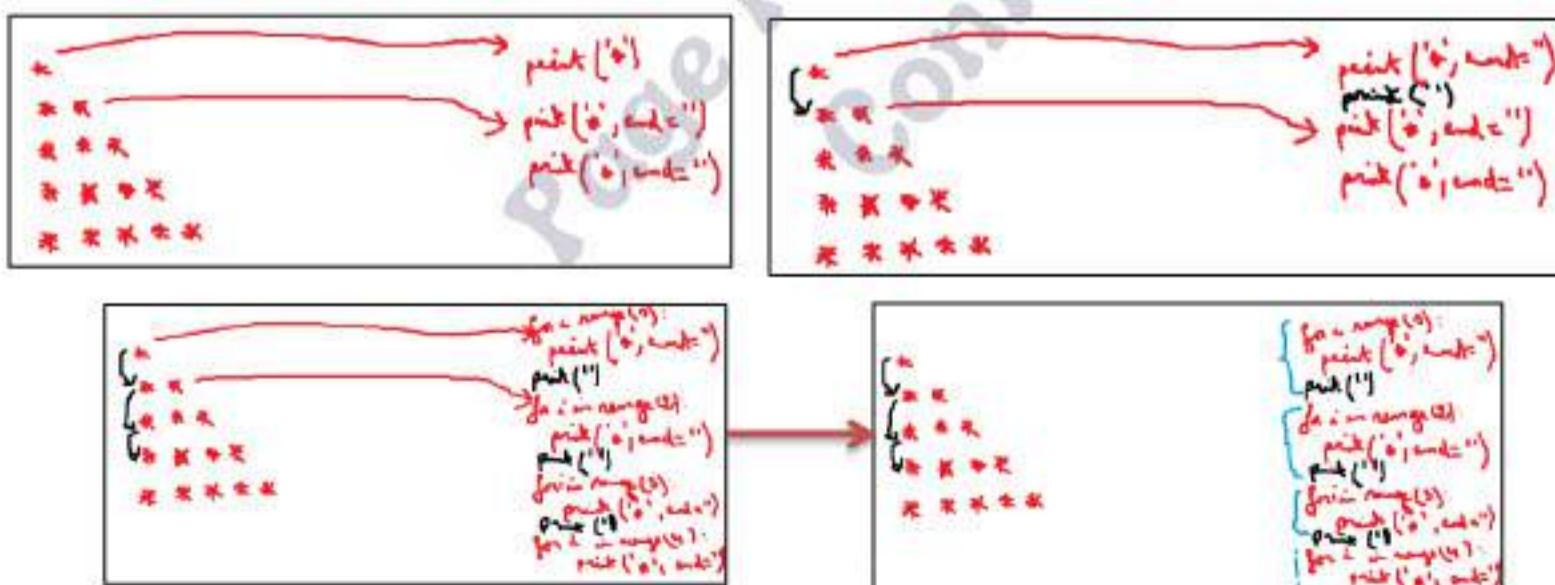
1 n = int (input ('Donner le nombre de ligne'))
2 ch = ''
3 for i in range (1,n+1):
4     ch = ch + '*'
5     print (ch)

Entrée
Donner le nombre de ligne5
*
**
***
****
*****

```

#Méthode 3 :

✓ Démarche et idées



Algorithme :

Début figure1

Ecrire('Donner le nombre des lignes')

Lire(n)

Pour i de 1 à n faire

Pour j de 0 à i-1 faire

Ecrire('*',end='')

Fin Pour

Fin Pour

Fin figure1

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
i	Entier	Compteur
j	Entier	Compteur

Python :

```

1 n = int (input ('Donner le nombre de ligne'))
2 for i in range (1, n+1):
3     for j in range (i):
4         print ('*', end = '')
5     print ('')

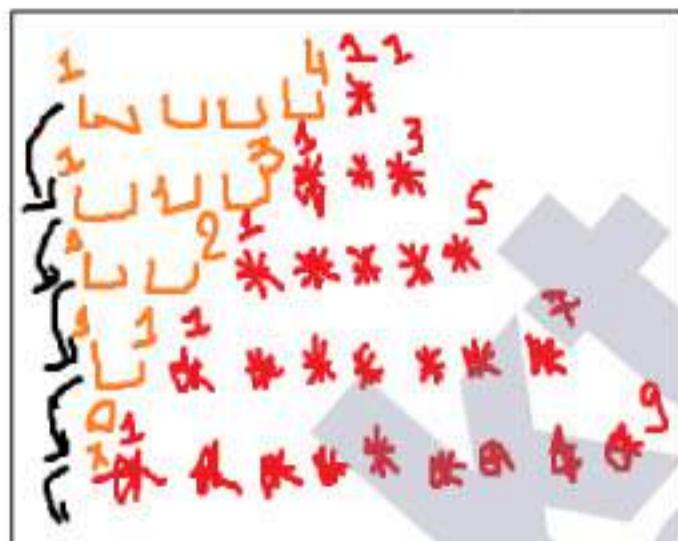
```

Console

```

Donner le nombre de ligne5
*
**
***
****
*****

```

**Figure 2****Algorithme :**

```

Début figure2
Ecrire('Donner le nombre des lignes')
Lire(n)
Pour i de 1 à n faire
    Pour j de 0 à n-i faire
        Ecrire(' ',end='')
    Fin Pour
    Pour k de 1 à 2*i -1 faire
        Ecrire('*',end='')
    Fin Pour
Fin Pour
Fin figure2

```

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
i	Entier	Compteur
j	Entier	Compteur
k	Entier	Compteur

Python :

```
1 n = int (input ('Donner le nombre de ligne'))
2 for i in range (1, n+1):
3     for j in range (1, (n-i)+1):
4         print (' ', end = '')
5     for k in range (1, (2*i -1)+1):
6         print ('*', end = '')
7     print ()
```

Console

```
Donner le nombre de ligne5
*
***
*****
*****
*****
```

5. Chapitre 5 : Les tableaux

Un tableau unidimensionnel (ou vecteur) est une structure de données permettant de ranger (regrouper) un nombre fini d'éléments de même type.

Un vecteur/ tableau est caractérisé par :

- ❖ Son nom (un identificateur unique. Exemple : T, V, U, ...)
- ❖ Sa taille (nombre d'éléments. Exemple : 20, 30, 100, ...)
- ❖ Son type (le type des éléments qu'il contient. Exemple : entier, réel, caractères, ...)

Syntaxe en algorithme et en python :

En algorithme :

Tableau de déclaration de nouveau type (T.D.N.T)

Type
Nom_de_type =tableau de taille de type

Tableau de déclaration d'objet(T.D.O)

Objet	Type
Nom_de_tableau	Nom_de_type

En python :

Les deux méthodes les plus couramment utilisées pour déclarer un tableau en Python sont list et numpy module.

Déclaration de Tableau comme list :

T=[] → initialisation d'un tableau (list) vide

Ou

T=[0] *taille → initialisation d'un tableau (list) remplir avec zéro

Déclaration de tableau avec le module numpy

```
from numpy import *
T=array([type_de_donne] *taille)
```

i. Définition d'une liste

Une liste est une structure de données qui contient une série de valeurs. Python autorise la construction de liste contenant des valeurs de types différents (par exemple entier et chaîne de caractères), ce qui leur confère une grande flexibilité. Une liste est déclarée par une série de valeurs (n'oubliez pas les guillemets, simples ou doubles, s'il s'agit de chaînes de caractères) séparées par des virgules, et le tout encadré par des crochets. En voici quelques exemples :

```
1 T=[1, 'hello', 1.4]
2 print(T)
```

```
>>> %Run tfsd.py
[1, 'hello', 1.4]
```

ii. Utilisation des listes

On appelle un élément de cette liste d'après son numéro (ou indice) pris dans l'ordre en commençant par 0 pour le premier élément.

```
1 T=[1, 'hello', 1.4]
2 print(T[1])
```

```
>>> %Run tfsd.py
hello
```

Pour connaître le nombre éléments d'une liste, on dispose de la fonction `len()` :

```
1 T=[1, 'hello', 1.4]
2 print(len(T))
```

```
>>> %Run tfsd.py
```

```
3
```

Comme pour les chaînes de caractères et avec la même syntaxe, il est possible de ne sélectionner qu'une partie de liste. Ne pas oublier que le premier indice vaut 0.

```
1 T=[1, 'hello', 1.4]
2 print(T[1:3])
```

```
>>> %Run tfsd.py
```

```
['hello', 1.4]
```



Souvenez-vous, en programmation, Dans les tableaux on ne peut pas stocker que des données de même type.

iii. Opération sur les listes

Comme pour les chaînes de caractères, il est possible de concaténer des listes, c'est à dire de les juxtaposer avec la fonction `"+"`.

```
1 a = [1, 2, 3]
2 b = [2, 3, 4]
3 listeconcat = a + b
4 print(listeconcat)
```

```
>>> %Run tfsd.py
```

```
[1, 2, 3, 2, 3, 4]
```

Il est possible d'ajouter des éléments à la fin d'une liste avec la fonction `append()`.

```

4 listeconcat.append(9)
5 print(listeconcat)

>>> %Run tfsd.py
[1, 2, 3, 2, 3, 4, 9]

```

iv. Définition d'un tableau

Un tableau est une liste ordonnée de nombres uniquement (flottants, entiers, complexes voire booléens codés en binaire). Les tableaux sont accessibles à partir de la librairie Numpy qui doit être préalablement chargée. Pour des raisons de simplicité évidentes.

Il est possible de construire un tableau (= "array") de manière explicite. Comme les listes, il s'agit d'un nouveau type de variable.

b. Remplissage d'un tableau

```

1 elet = int (input ('Donner un entier'))
2 T.append(elet)

Kernel
T.append(elet)
NameError: name 'T' is not defined

```

Il faut initialiser le tableau T à vide d'abord.

```

1 T = []
2 print (T)
3 elet = int (input ('Donner un entier'))
4 T.append(elet)
5 print (T)

Kernel
[]
Donner un entier12
[12]

```

```

1 T = []
2 print (T)
3 elet = int (input ('Donner un entier'))
4 T.append(elet)
5 print (T)
6 elet = int (input ('Donner un entier'))
7 T.append(elet)
8 print (T)

Console:
[]
Donner un entier12
[12]
Donner un entier15
[12, 15]

```

En cas général, pour remplir des tableaux de taille quelconque, on va utiliser la boucle Pou

c. Autre méthode du remplissage du tableau

La méthode append() ou la concaténation de deux tableaux sont des méthodes pratiques, en algorithme on remplit un tableau de la manière suivante :

Debut remplissage

Répéter

Ecrire("Donner la taille du tableau")

Lire(n)

jusqu'à $n \geq 0$ et $n \leq 20$

Pour i de 0 à $n-1$ faire

Ecrire("donner $T[i]$ ")

Lire($T[i]$)

Fin Pour

Fin remplissage

TDO

Objet	Type/Nature	Rôle
n	Entier	Donné
T	tab	Tableau d n entier

i	Entier	Compteur
---	--------	----------

TDNT

Type
tab = Tableau de 20 entiers

En pratique, la méthode de « .append » est différente à celle en théorique, donc on a 2 autres méthodes en pratique de remplissage d'un tableau :

➤ **Remplissage d'un tableau (déclaration de la liste en 0) :**

Pour s'approcher d'une traduction similaire à la méthode théorique, on va faire une affectation directement sur T[i] lors du saisie.

Mais, on doit donner des valeurs de 0 (ou autre) à tous les éléments du tableau car, on ne peut affecter une valeur à T[i] qui n'est pas encore définie.

```

1 n = int (input ('Donner le nombre des éléments'))
2 T= [0]*n
3 print(T)
4 for i in range (n):
5     print ('Donner un élément numéro ',i)
6     T[i] = int (input ())
7     print (T)

Console
Donner le nombre des éléments3
[0, 0, 0]
Donner un élément numéro 0
12
[12, 0, 0]
Donner un élément numéro 1
15
[12, 15, 0]
Donner un élément numéro 2
19
[12, 15, 19]
```

Remplissage d'un tableau (avec la bibliothèque numpy)

```

1 from numpy import *
2 n = int (input ('Donner le nombre des éléments'))
3 T = array ([int()] *n)
4 print (type(T))
5 for i in range (n):
6     T[i] = int (input('Donner un entier'))
7 print (T)

onsole
Donner le nombre des éléments3
<class 'numpy.ndarray'>
Donner un entier12
Donner un entier15
Donner un entier14
[12 15 14]

```

NumPy est une bibliothèque dans le langage de programmation Python

La structure de données principale dans NumPy est le **ndarray**, qui est un nom abrégé pour un tableau à N dimensions. Lorsque vous travaillez avec NumPy, les données d'un ndarray sont simplement appelées un tableau. Il s'agit d'un tableau de taille fixe en mémoire qui contient des données du même type, telles que des nombres entiers ou des chaînes.

Remarque : x = int() c'est-à-dire x = 0 et ch = str() c'est-à-dire ch = "" (chaîne vide)

Ou on peut écrire aussi :

```

1 import numpy as np
2 n = int (input ('Donner le nombre des éléments'))
3 T = np.zeros((n))
4 print (T)
5 print (type(T))
6 for i in range (n):
7     T[i] = int (input('Donner un entier'))
8 print (T)

onsole
Donner le nombre des éléments3
[0. 0. 0.]
<class 'numpy.ndarray'>
Donner un entier12
Donner un entier14
Donner un entier15
[12. 14. 15.]

```

Exercice : Somme des carrées des chiffres

Ecrire un programme qui permet de remplir un tableau T1 de N1 entiers naturels de trois chiffres ($1 < N1 < 100$) et on se propose de remplir et afficher un tableau T2 de la façon suivante :

$T2[i] = \text{somme des carrées des chiffres de } T1[i]$

Exemple : $T1[2] = 254$ alors $T2[2] = 2^2 + 5^2 + 4^2 = 45$

Correction de l'exercice

Algorithme :

Début Ex1_tableau

Répéter

 Ecrire('Donner la taille du tableau')

 Lire(N1)

 Jusqu'à $1 < N1 < 100$

 Pour i de 0 à N1-1 faire

 Répéter

 Lire(T1[i])

 Jusqu'à $100 \leq T1[i] \leq 999$

 Fin Pour

 Pour i de 0 à N1-1 faire

 u = T1[i] mod 10

 d = (T1[i] mod 100) div 10

 c = T1[i] div 100

$T2[i] = u^2 + d^2 + c^2$

 Fin Pour

 Ecrire(T2)

Fin Ex1_tableau

TDO

Objet	Type/Nature	Rôle
N1	Entier	Taille du tableau
T1	tab1	Tableau des entiers
i	Entier	Compteur
T2	tab2	Tableau des entiers

TDNT

Type
tab1=Tableau de 100 entiers naturels de 3 chiffres
tab2=Tableau de 100 entiers naturels

Python :

```

1 test=True
2 while test:
3     N1=int(input('Donner la taille du tableau'))
4     if 1<=N1<=100:
5         test=False
6 T1=[0]*N1
7 for i in range(N1):
8     test1=True
9     while test1:
10        T1[i]=int(input('Donner T['+str(i)+']'))
11        if 100<=T1[i]<=999:
12            test1=False
13 T2=[0]*N1
14 for i in range(N1):
15    T2[i]=(T1[i]%10)**2+((T1[i]%100)//10)**2+(T1[i]//100)**2
16 for i in range(N1):
17    print(T2[i],end=' | ')
18

```

d. Le maximum / minimum dans un tableau

Pour chercher le maximum ou le minimum dans un tableau il faut d'abord créer une variable maxi (voir mini) et on lui affecte la valeur de la première case, puis parcourir le tableau case par case et vérifier à chaque fois si $T[i] > \text{maxi}$ (voir $T[i] < \text{mini}$) ou non, si c'est le cas on change la valeur de la variable maxi (mini) par $T[i]$

```

1 n = 5
2 T= [5,12,7,19,3]
3 maxi = T[0]
4 for i in range (1, n):
5     if T[i] > maxi :
6         maxi = T[i]
7 print ('Le maximum est',maxi)

```

```

1 n = 5
2 T= [5,12,7,19,3]
3 mini = T[0]
4 for i in range (1, n):
5     if T[i] < mini :
6         mini = T[i]
7 print ('Le minimum est',mini)

```

e. Exercice longueur maximale des chaînes

Exercice : Tableau des chaînes

Elaborer un programme qui permet de remplir un tableau T par n chaînes de caractères ($4 \leq n \leq 20$), cherche et affiche la longueur de la chaîne la plus longue puis affiche toutes les chaînes ayant cette longueur

Correction de l'exercice

Algorithme :

Début Ex2_tableau

Répéter

Ecrire('Donner la taille du tableau')

Lire(n)

Jusqu'à $4 \leq N1 \leq 20$

Pour i de 0 à n-1 faire

Lire(T[i])

Fin Pour

maxi \leftarrow long(T[0])

Pour i de 1 à n-1 faire

si long(T[i]) > maxi alors maxi \leftarrow long(T[i])

Fin Si

```

Fin Pour
Ecrire(maxi)
Pour i de 0 à n-1 faire
    Si long(T[i])=maxi alors Ecrire (T[i])
Fin Si
Fin Pour
Fin Ex2_tableau

```

TDO

Objet	Type/Nature	Rôle
n	Entier	Taille du tableau
T	tab	Tableau de n chaînes de caractères
i	Entier	Compteur
maxi	Entier	La longueur maxiamel

TDNT

Type
tab=Tableau de 20chaîne de caractères

Python :

```

1 while True:
2     n = int (input ('Donner le nombre des éléments'))
3     if n >4 and n <20 :
4         break
5 T=[]
6 for i in range (1,n+1):
7     print('Donner la chaîne n°', i)
8     ch = input('')
9     T.append(ch)
10 print(T)
11 maxi = len(T[0])
12 for i in range (1,n):
13     if len(T[i]) >maxi :
14         maxi = len(T[i])
15 for i in range (n):
16     if len(T[i]) == maxi:
17         print (T[i])

```

En utilisant la bibliothèque numpy

```

1 from numpy import *
2 #saisie taille du tableau
3 test=True
4 while test:
5     n=int(input('donner la taille du tableau:'))
6     if 4<n<20:
7         test=False
8 #remplir tableau T
9 T=array([int]*n)
10 for i in range(n):
11     T[i]=input('donner un mot dans la case {}:'.format(i+1))
12 #rechercher le maximum longueur
13 maxi=len(T[0])
14 for i in range(1,n):
15     if maxi<len(T[i]):
16         maxi=len(T[i])
17 #afficher
18 for i in range(n):
19     if maxi==len(T[i]):
20         print(T[i])
21

```

```

donner la taille du tableau:5
donner un mot dans la case 1:hello
donner un mot dans la case 2:welcome
donner un mot dans la case 3:nice
donner un mot dans la case 4:hey
donner un mot dans la case 5:like it
welcome
like it

```

Exercice

Soit le tableau T suivant :

10	7	9	7	10	6	7	4	8	8
----	---	---	---	----	---	---	---	---	---

Pour chaque élément de T on ne garde que sa première occurrence et on remplace les autres par 0.

10	7	9	0	0	6	0	4	8	0
----	---	---	---	---	---	---	---	---	---

Pour regrouper les éléments restant au début du tableau T.

10	7	9	6	4	8	0	0	0	0
----	---	---	---	---	---	---	---	---	---

Ecrire un programme Python qui fait le traitement ci-dessus pour un tableau T de n ($2 < n < 20$) entiers positifs non nuls et détermine et affiche le nombre d'éléments différents de T.

Solution :

Saisie et remplir le tableau T

```

1 from numpy import *
2 #saisie taille du tableau
3 test=True
4 while test:
5     n=int(input('donner la taille du tableau:'))
6     if 2<=n<=20:
7         test=False
8 #remplir tableau T
9 T=array([int]*n)
10 for i in range(n):
11     test=True
12     while test:
13         T[i]=int(input('donner un entier positif dans la case {}:{}.'.format(i+1)))
14         if T[i]>0:
15             test=False
16

```

Garder et regrouper le tableau T dans T1 :

```

17 #garder la premeier occurence
18 for i in range(n-1):
19     for j in range(i+1,n):
20         if T[i]==T[j]:
21             T[j]=0
22
23 #regrouper
24 T1=array([0]*n)
25 j=0
26 for i in range(n):
27     if T[i]>0:
28         T1[j]=T[i]
29         j=j+1
30 print(T1)

```

6. Chapitre 6 : Sous-Programme

L'analyse modulaire a pour objectif de décomposer le problème en sous problèmes (sous programmes, les sous programmes en d'autres sous programmes plus simples et ainsi de suite jusqu'à ce qu'on obtienne des sous programmes élémentaires. On distingue deux types de sous programmes : Les procédures et les fonctions

Les *sous-programmes* sont ainsi le moyen pour le programmeur de définir ses propres instructions et opérations. On appelle généralement *fonction* un sous-programme qui retourne un résultat. C'est le cas de la fonction cube qui sert à calculer et retourner la racine cubique d'un entier donné. On appelle *procédure* un sous-programme qui ne retourne pas de valeur. Une procédure se comporte donc comme une instruction.

Avantages des sous programmes.

- Les sous programmes permettent de ne pas répéter plusieurs fois une même séquence d'instructions au sein du programme.
- Une organisation meilleure du programme
- Facilite la maintenance et la recherche des bugs dans un programme
- Un sous-programme peut être intégré à un autre programme, ou elle pourra être rangée dans une bibliothèque d'outils ou il pourra être utilisé par n'importe quel programme.

Un sous-programme peut-être une procédure ou une fonction.

a. Syntaxe des sous-programmes

i. Procédure

Une procédure est un sous-programme qui permet la résolution d'un problème donné et qui peut avoir plusieurs résultats à partir d'un ou plusieurs paramètres donnés

Déclaration d'une procédure

En général la syntaxe de la procédure est :

Procédure nom_procédure (Mode de passage + Liste des paramètres formels : Type)

Début



Fin

```

1 def nom_procédure (Paramètres):
2     '''instruction'''
3     '''instruction'''
```

Exemple :

Procédure Saisie (@ n : entier)

Début

Répéter

Ecrire(" Donner un entier")

Lire (n)

Jusqu'à (n>=0) et (n<=20)

Fin

➤ Mode de passage : (Passage par valeur ou passage par variable) :

Si le contenu d'une variable globale dans une procédure change en passant par le traitement, on écrit « @ » avant la variable dans l'entête de la déclaration de la procédure (on appelle ça passage par variable) sinon on n'écrit rien (on appelle ça passage par valeur)

La notion du mode de passage est valable seulement pour les procédures.

Passage par valeur

C'est le processus par défaut :

Lorsqu'un paramètre est passé par valeurs, le compilateur crée une variable locale du même type et lui affecte la valeur du paramètre. Ainsi une modification dans le corps de la procédure n'affecte que cette variable locale.

la modification ne se répercutant pas à l'extérieur de la procédure sur le paramètre passé.

Passage par variable

Dans ce mode de passage, le programme appelé et le programme appelant font un échange de données. En effet, toute modification de la valeur d'un paramètre au sein du programme appelé doit être communiquée au programme appelant. Dans ce mode de passage, on précède le paramètre à transmission par variable par le mot @.

Appel d'une procédure :

On appelle une procédure dans le programme principal au-dessus de la déclaration du sous-programme ou dans un autre sous-programme.

Exemple :

Saisie (n)

En général la syntaxe de l'appel d'une procédure est :

nom_procedure (paramètres effectifs)

Exemple Python :

```

1 #Déclaration du sous-programme
2 def Saisie():
3     while True:
4         n = int (input ('Donner un entier'))
5         if n >=0 and n <=20 :
6             break
7 #Programme principal
8 #Appel de la procédure
9 Saisie ()
```

ii. Fonction

On appelle un algorithme en tant que fonction, tout traitement qui a pour objectif le calcul ou la détermination d'un seul résultat, retourne une seule variable comme résultat final.

C'est-à-dire une fonction est un sous-programme qui retourne un résultat unique de type simple (entier, réel, booléen, caractère, chaîne de caractères).

Déclaration d'une fonction

En général la syntaxe de la fonction est :

Fonction nom_fonction (Liste des paramètres formels : Type) : Type du résultat retourné

Début

..... } Traitement
.....

Retourner nom_variable contenant le résultat calculé

Fin

```
1 def nom_fonction (Paramètres):  
2     '''instruction'''  
3     '''instruction'''  
4     return résultat
```

Appel d'une fonction :

On appelle une fonction dans le programme principal au-dessus de la déclaration du sous-programme ou dans un autre sous-programme.

Exemple

p ← somme (n)

Écrire (p)

Remarque : p est une autre variable qui reçoit le résultat de la fonction

En général la syntaxe de l'appel d'une fonction est :

p ← nom_fonction (paramètres effectifs)

En pratique, on peut utiliser des procédures en tant que fonction, cela est nécessaire lorsqu'on va utiliser le résultat de la procédure dans un autre sous-programme en suite.

Exemple :

```

1 def Saisie():
2     test = False
3     while not (test) :
4         n = int (input ('Donner un entier'))
5         if n>= 0 and n <= 20:
6             test = True
7     return n
8 def Affichage_diviseurs ():
9     s = 0
10    for i in range (1, n +1):
11        if n % i == 0:
12            print (i)
13 n = Saisie ()
14 Affichage_diviseurs()

```

Ici, le sous-programme `Affichage_diviseurs` nécessite la variable `n` dans son fonctionnement donc, elle doit être définie comme variable globale dans le programme principal.

- *C'est mieux de définir le sous-programme `Saisie` comme fonction pour retourner `n` facilement dans le programme principal.*

Il existe une autre méthode peu utilisée, c'est d'utiliser l'algorithme de saisie ou autre algorithme en tant que procédure et non pas une fonction, cela se fait en ajoutant la variable globale devant l'instruction `global` en python, c'est-à-dire cette sera globale et visible pour tout le programme.

Exemple :

```

1 def Saisie () :
2     global n
3     test = False
4     while not (test) :
5         n = int (input ('Donner un entier'))
6         if n >= 0 and n <= 20:
7             test = True
8 def Affichage_diviseurs () :
9     s = 0
10    for i in range (1, n +1):
11        if n % i == 0:
12            print (i)
13 Saisie ()
14 Affichage_diviseurs ()

```

★ Paramètre formel et paramètre effectif :

Exemple :

{Déclaration}

Fonction Somme (**x** : entier) : entier

Début

s \leftarrow 0

Pour i de 1 à **x** faire

Si **x** mod i = 0 alors

s \leftarrow s + 1

Fin Si

Fin Pour

Retourner s

Fin

{Appel}

Ecrire ("Donner un entier")

Lire (n)

$p \leftarrow \text{Somme}(n)$

Écrire (p)

Ou on peut afficher la somme comme ça : Écrire (Somme (n))

Exemple Python :

```

1 def nb_div (x):
2     s = 0
3     for i in range (1, x+1):
4         if x % i == 0 :
5             s = s + 1
6     return s
7
8 n = int (input('Donner un entier n'))
9 print (nb_div (n))
10 m = int (input('Donner un entier m'))
11 print (nb_div (m))

```

➤ Paramètres effectifs / Paramètres formels :

On appelle x paramètre formels c'est-à-dire il prend un nom de variable quelconque lors de la déclaration du sous-programme mais après dans l'appel de la fonction, le programme fonctionne réellement avec m et n qui sont des paramètres effectifs.

La notion des paramètres formels et effectifs est valable aussi pour les procédures.

➤ Variable globale et variable locale :

- Variable locale

Une variable locale est une variable déclarée dans un sous-programme. Elle n'est accessible que depuis ce sous-programme.

Remarque : Un paramètre est l'équivalent d'une variable locale initialisée lors de l'appel du sous-programme alors qu'une variable locale est une variable qui est initialisée dans le sous-programme et ne sera pas accessible de l'extérieur de ce sous-programme.

- **Variable globale**

Une variable globale est une variable qui est potentiellement accessible de plusieurs sous-programmes. Une variable est donc dite globale si sa portée inclut plusieurs sous-programmes.

Fiche récapitulative

- La définition d'une procédure (ou d'une fonction) se fait toujours dans la partie déclarative du programme principal.
- Le programme principal appelle (utilise) les sous programmes définis dans sa partie déclarative : il est dit programme appelant et les sous programmes sont dits programmes appelés.
- Les paramètres utilisés au moment de la définition d'un sous-programme sont appelés des paramètres formels (fictifs).
- Les paramètres utilisés au moment de l'appel d'un sous-programme par le programme principal (ou par un autre sous-programme) sont appelés des paramètres effectifs (réels).
- Les variables locales sont des variables déclarées et utilisées uniquement par le sous-programme où elles sont déclarées.
- Les variables globales sont des variables déclarées par le programme principal et utilisées par n'importe quel sous-programme.
- La substitution des paramètres effectifs aux paramètres formels s'appelle transmission des paramètres.
- L'appel d'une procédure respecte la syntaxe suivante : Nom_procedure (parameffect 1,parameffect 2, ..., parameffect n).
- Les paramètres d'un sous-programme ne sont pas des variables, il ne faut donc pas les déclarer.
- L'appel d'une fonction peut apparaître partout :

- + Dans une affectation : `x = max (a, b) ;`
- + Dans une comparaison : `if pair (a) : ...`
- + Dans un affichage : `print ('Le maximum est : ', max (a, b))`
- Il est possible (mais déconseillé) d'utiliser l'identificateur du paramètre d'une fonction pour nommer une variable globale.
- Pour le choix d'un nom de fonction il faut respecter les mêmes règles que celles pour les noms de variables.
- Le type de fonction est le type du résultat retourné (Entier, réel, booléen, etc.)
- L'instruction `return` sert à retourner la valeur du résultat
- Une fonction peut avoir de 0 à N paramètres
- `Param1, param2, ...,` sont appelés paramètres(Arguments) : ce sont des variables qui
- Permettent à la fonction de communiquer avec l'extérieur.
- Ces paramètres déclarés lors de la définition de la fonction sont appelés paramètres formels.

Applications avec sous-programme :

Exercice d'application 1 :

Ecrire une fonction nommée `maximum` qui calcule et renvoie le plus grand de deux entiers A et B.

Solution en algorithme :

Fonction maximum (A : entier, B : entier) : entier

```

Début
Si A>B alors
    Max ← A
Sinon
    Max ← B
Fin Si
Retourner max
Fin

```

L'APPEL D'UNE FONCTION :

Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom suivi des paramètres effectifs dans le programme principal.

Le résultat d'une fonction étant une valeur, devra être affecté à une variable ou être utilisé dans une expression, calcul, affichage, test, ... etc.

Appel de la fonction maximum qui renvoie le maximum deux nombres.

Algorithme maximum_deux_nombre

Début :

Ecrire ("saisir une valeur de A ") Lire(A)

Ecrire ("saisir une valeur de B ") Lire(B)

//Appel de la fonction maximum()

Max ← maximum (A, B)

Ecrire ("Le maximum est : ", max)

Fin

Exercice d'application 2 :

Appel de la procédure permute qui prend en paramètres deux nombres A et B de type entier puis afficher le résultat de permutation.

Solution en algorithme :

Procédure permute (@A : entier, @B : entier)

Début

Vide \leftarrow A

A \leftarrow B

B \leftarrow Vide

Ecrire ("La nouvelle valeur de A : ", A)

Ecrire ("La nouvelle valeur de B : ", B)

Fin

Contrairement à l'appel d'une fonction, on ne peut pas affecter la procédure appelée ou l'utiliser dans une expression. L'appel d'une procédure est une instruction autonome.

Algorithme permuter_deux_nombre

//Algorithme principal

Début

Ecrire ("saisir une valeur de A ") Lire(A)

Ecrire ("saisir une valeur de B ") Lire(B)

//Appel de la procédure permuter ()

Permuter (A, B)

Fin

Exercice :

Élaborer un programme qui permet de saisir un vecteur T par n chaînes de caractères ($4 < n < 20$), cherche et affiche la longueur de la chaîne la plus longue puis affiche toutes les chaînes ayant cette longueur.

Correction Théorique :

Procédure Saisie (@n : entier)

Début

Répéter

Ecrire ("Donner le nombre des éléments")

Lire (n)

Jusqu'à $n > 4$ et $n < 20$

Fin

Procédure Remplissage (n : entier, @T : Tab)

Début

Pour i de 0 à $n - 1$ faire

Ecrire ("Donner T[", "i, ", "]")

Lire (T[i])

Fin Pour

Fin

T.D.O.L

Objet	T/N	Rôle
i	entier	compteur

Fonction Lmax (n : entier, T : Tab) : entier**Début**

```
maxi ← long (T[0])
```

Pour i de 1 à n-1 faire

Si long(T[i]) > maxi alors

```
    maxi ← T[i]
```

Fin Si

Fin Pour

Retourner maxi

Fin

T.D.O.L

Objet	T/N	Rôle
i	entier	compteur
maxi	entier	La longueur maximale des chaines du tableau

Procédure Affichage (n : entier, T : Tab)**Début**

Pour i de 0 à n-1 faire

Si long (T[i]) = Lmax (n,T) alors

Ecrire(T[i])

Fin Si

Fin Pour

Finv

T.D.O.L

Objet	T/N	Rôle
i	entier	compteur
Lmax	fonction	Déterminer la longueur maximale des chaînes

Programme principal**Début**

Saisie (n)

Remplissage (n,T)

Affichage (n,T)

Fin**T.D.N.T**

Type
Tab = tableau de 19 chaînes

T.D.O.G

Objet	T/N	Rôle
n	entier	donnée
T	Tab	Tableau de chaînes
Saisie	procédure	Saisie du nombre des éléments
Remplissage	procédure	Remplissage du tableau
Affichage	procédure	Affichage des chaînes ayant la longueur maximale

Python:

```

1 def Saisie():
2     test = False
3     while not (test):
4         n = int (input('donner le nombre des éléments'))
5         if n > 4 and n < 20 :
6             test = True
7     return n
8 def Remplissage():
9     for i in range (n):
10        T[i] = input ('Donner T['+str(i)+']')
11    return T
12 def Lmax (n,T):
13     maxi = len (T[0])
14     for i in range (1,n):
15         if len(T[i]) > maxi:
16             maxi = T[i]
17     return maxi

```

```

18 def Affichage():
19     for i in range (n):
20         if len (T[i]) == Lmax (n,T):
21             print (T[i])
22
23 n = Saisie ()
24 T = [0]*n
25 T = Remplissage()
26 print ('Les chaines qui ont la même longueur maximale sont')
27 Affichage ()

```

7. Chapitre 7 : Les algorithmes de tri

7.1. Introduction

Un algorithme de tri est une suite finie d'instructions servant à réordonner une séquence d'éléments suivant un critère fixé à priori (ordre croissant, ordre décroissant).

Par exemple si on a un tableau T de taille n=6

20	3	50	12	35	7
1	2	3	4	5	6

Trier le tableau en ordre croissant (de plus petit au plus grand) donne le résultat suivant :

3	7	12	20	35	50
1	2	3	4	5	6

Trier le tableau en ordre décroissant (de plus grand au plus petit donne le résultat suivant :

50	35	20	12	7	3
1	2	3	4	5	6

Remarque

On peut trier autres types que les entiers. Il suffit de disposer d'un domaine de valeurs muni d'une relation d'ordre total. On peut donc trier des caractères, des mots en ordre alphabétique

7.2. Les algorithmes de tri

On distingue plusieurs méthodes de tri (sélection, bulles, insertion, Shell...). La plupart des algorithmes de tri sont basés sur deux opérations essentielles : La comparaison et l'échange (permutation).

7.2.1. Tri par sélection

Cette méthode de tri consiste à :

- Se pointer à la 1^{ère} case du tableau T et de parcourir la totalité du tableau pour repérer l'indice de la première position du minimum.
- Comparer ce minimum avec T [1]. S'ils sont différents on les permute.
- Le sous tableau de T allant de 2 à n est à priori non trié, on applique l'étape 1 et 2 et ainsi de suite jusqu'à l'avant dernier élément (n-1).

Exemple :

Soit un tableau T contenant les dix éléments suivants :

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

Etape 1 : Parcourir la totalité du tableau pour repérer le minimum (indice de la première position du minimum) et le comparer avec T [1].

T :	12	10	0	-5	8	12	-2	2	40	-1
	1	2	3	4	5	6	7	8	9	10
↑ ↑ Indice du minimum										
T [1] ↔ T [4] alors permutation										

On obtient :

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

Le sous tableau allant de 2 à n est à priori non trié, on applique l'étape 1 et 2 et ainsi de suite jusqu'à l'avant dernier élément (n-1).

Etape 2 :

A horizontal number line with tick marks at every integer from -5 to 10. The value -5 is highlighted with a cyan box. An arrow points to the value 2 , and another arrow points to the value -2 .

T [2] \leftrightarrow T [7] alors permutation

On obtient :

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

Etape3 :

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

Indice du minimum

T [3] ↔ T [10] alors permutation

On obtient :

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

Etape 4 :

T :

-5	-2	-1	12	8	12	10	2	40	0
----	----	----	----	---	----	----	---	----	---

Indice du minimum

$T[4] \leftrightarrow T[10]$ alors permutation

On obtient :

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

Etape 5 :

On obtient :

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

Etape 6 :

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

↑
↑
Indice du minimum

T [6] ↔ T [8] alors permutation

On obtient :

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

Etape 7 :

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

↑
↑
Indice du minimum

T [7] = T [7] alors pas de permutation

On obtient :

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

Etape 8 :

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

↑
↑
Indice du minimum

T [8] = T [8] alors pas de permutation

On obtient :

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

Etape 9 :

T :	-5	-2	-1	0	2	8	10	12	40	12
	1	2	3	4	5	6	7	8	9	10
T [9] ↔ T [10] alors permutation										↑ Indice du minimum

On obtient :

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

Remarque

On est arrivé à l'élément numéro $n-1$, alors arrêt du traitement. Nous n'avons pas besoin de traiter le dernier élément, puisque si les neuf premiers éléments sont triés alors automatiquement le dernier sera le plus grand et par conséquent il se trouve à la bonne position.

La procédure de tri sélection est donnée comme suit :

Algorithme :

Procédure tri_sélection (@T :tab ; n :entier)

Début

Pour i de 0 à $n-2$ faire

 Posmin \leftarrow i

 Pour j de $i+1$ à $n-1$ faire

 Si $T[\text{Posmin}] > T[j]$ alors

 Posmin $\leftarrow j$

```

Fin si

Fin Pour

Vide ← T[i]

T[i] ← T[Posmin]

T[Posmin] ← Vide

FinPour

Fin

```

Python :

```

1 def tri_selection (T,n):
2     for i in range(n-1):
3         posmin=i
4         for j in range(i+1,n):
5             if T[posmin]>T[j]:
6                 posmin=j
7             vide=T[i]
8             T[i]=T[posmin]
9             T[posmin]=vide

```

7.2.2. Tri à Bulles

Cette méthode de tri consiste à :

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

Exemple :

Soit un tableau T contenant les dix éléments suivants :

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

Soit une variable booléenne test initialisée à vrai et qui devienne faux à chaque permutation.

Passage 1 :

T[1] > T[2] alors
permutation

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

T[2] > T[3] alors
permutation

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

T[3] > T[4] alors
permutation

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

T[4] > T[5] alors
permutation

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

T[5] = T[6] alors pas
de permutation

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

T[6] > T[7] alors
permutation

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

T[7] > T[8] alors
permutation

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

T[8] < T[9] alors pas
de permutation

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

T[9] > T[10] alors
permutation

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

Puisqu'on a atteint la fin du tableau et le contenu de la variable test est Faux, alors on doit recommencer un nouveau passage et ainsi de suite jusqu'à ce qu'on fasse un passage complet du tableau sans modifier le contenu de test (Vrai).

La procédure de tri sélection est donnée comme suit :

Algorithme

Procédure tri_bulles (@T :tab ; n : entier)

Début

Répéter

 Tri \leftarrow Vrai

 Pour i de 0 à n-2 faire

 Si T[i] > T[i+1] alors

 Vide \leftarrow T[i]

 T[i] \leftarrow T[i+1]

 T[i+1] \leftarrow Vide

 Tri \leftarrow Faux

 Fin Si

 Fin Pour

Jusqu'à Tri = Vrai

Fin

Python

```

1 def tri_bulle (T,n):
2     test=True
3     while test:
4         tri=True
5         for i in range(n-1):
6             if T[i]>T[i+1]:
7                 vide=T[i]
8                 T[i]=T[i+1]
9                 T[i+1]=vide
10                tri=False
11            if tri == True:
12                test=False

```

Exercice d'application : Bac pratique 2009

REPUBLIQUE TUNISIENNE MINISTERE DE L'EDUCATION ET DE LA FORMATION EXAMEN DU BACCALAUREAT SESSION 2009	Sections : Math. + Tech. + Sc.Exp. EPREUVE PRATIQUE D'INFORMATIQUE Date : mardi 19 mai 2009 à 15h 30 DUREE : 1 h COEFFICIENT : 0,5
--	--

Important :

1. Une solution modulaire au problème est exigée.
2. Enregistrez au fur et à mesure votre programme dans le dossier Bac2009 se trouvant sur la racine C:\ en lui donnant comme nom votre numéro d'inscription.

Soit un tableau T de N lettres minuscules ($6 \leq N \leq 100$), et soient D et N deux entiers qui répondent aux conditions suivantes :

1. D est un entier diviseur de N strictement supérieur à 1.
2. M est un entier tel que $N = M \cdot D$.

On se propose de trier les D éléments des M blocs disjoints qui constituent le tableau T.

Ecrire un programme Pascal permettant de :

- lire les deux entiers N et D qui répondent aux conditions 1. et 2.
- remplir le tableau T par N lettres minuscules,
- trier dans l'ordre croissant, les éléments de chaque bloc du tableau T,
- afficher le tableau T après le tri.

Exemple :

Si $N=12$ et $D=3$ (donc $M=4$) et si les éléments du tableau T sont les suivants :

$T =$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr> <td>a</td><td>b</td><td>a</td><td>c</td><td>b</td><td>t</td><td>g</td><td>f</td><td>a</td><td>k</td><td>d</td><td>f</td> </tr> <tr> <td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td> </tr> </table>	a	b	a	c	b	t	g	f	a	k	d	f	1	2	3	4	5	6	7	8	9	10	11	12
a	b	a	c	b	t	g	f	a	k	d	f														
1	2	3	4	5	6	7	8	9	10	11	12														
	Bloc 1 Bloc 2 Bloc 3 Bloc 4																								

Ce tableau contient 4 blocs de 3 lettres chacun.

Après le tri des éléments de chacun des blocs, le tableau T sera égal à :

$T =$	<table border="1" style="border-collapse: collapse; width: 100%;"> <tr> <td>a</td><td>a</td><td>b</td><td>b</td><td>c</td><td>t</td><td>a</td><td>f</td><td>g</td><td>d</td><td>f</td><td>k</td> </tr> <tr> <td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td> </tr> </table>	a	a	b	b	c	t	a	f	g	d	f	k	1	2	3	4	5	6	7	8	9	10	11	12
a	a	b	b	c	t	a	f	g	d	f	k														
1	2	3	4	5	6	7	8	9	10	11	12														
	Bloc 1 Bloc 2 Bloc 3 Bloc 4																								

Correction de l'exercice : Bac pratique 2009

Remarque : Pour la correction de cet exercice, nous allons considérer que le premier élément dans le tableau T est d'indice 0 donc les boucles Pour vont commencer de 0 à N-1. Cela ne change rien pour la méthode si on veut considérer que le premier élément dans le tableau est d'indice 1 sauf que les boucles Pour vont commencer de 1 à N.

#Algorithme du Programme principal

Début

Saisie_N(N)

Saisie_D(N,D)

M ← N div D

Remplir(T,N)

Trier(T,N,M,D)

Affichage(T,N)

Fin

TDNT

Tab	Tableau de N caractères minuscules
-----	------------------------------------

TDOG

Objet	Type/Nature
N	Entier
D	Entier
M	Entier
T	Tab
Saisie_N	Procédure
Saisie_D	Procédure
Remplir	Procédure
Trier	Procédure

Affichage

Procédure

#Déclaration des sous programmes

Procédure Saisie_N (@n :entier)

Début

Répéter

Ecrire("Donner la taille du tableau")

Lire(N)

Jusqu'à $6 \leq N \leq 100$

Fin

Procédure Saisie_D (N :entier, @D :entier)

Début

Répéter

Ecrire("Donner une entier ")

Lire (D)

Jusqu'à $N \bmod D = 0$ et $D > 1$

Fin

Procédure Remplir (@T : Tab, N :entier)

Début

Pour i de 0 à N-1 faire

Répéter

Ecrire("T[",i,"] =")

Lire($T[i]$)

Jusqu'à "a" ≤ $T[i] \leq z$ "

Fin

TDOL

Objet	Type/Nature
i	Entier

#Dans la procédure Trier nous allons appliquer le principe du tri par sélection pour les M blocs du tableau.

Procédure Trier (@T : Tab, N,M,D : entier)

Début

deb ← 0

fin ← D

Pour i de 0 à M-1 faire

Pour j de deb à fin -2 faire

posmin ← j

Pour k de j+1 à fin-1 faire

si $T[k] < T[posmin]$ alors

posmin ← k

Fin si

vide ← T[j]

$T[j] \leftarrow T[posmin]$

$T[posmin] \leftarrow \text{vide}$

Fin Pour

deb \leftarrow fin

fin \leftarrow fin +D

Fin Pour

Fin

TDOL

Objet	Type/Nature
deb	Entier
fin	Entier
i	Entier
j	Entier
k	Entier
posmin	Entier
vide	Caractère

Procédure Affichage (@T :Tab, N :entier)

Début

Pour i de 0 à N-1 faire

Ecrire(T[i])

Fin Pour

Fin

TDOL

Objet	Type/Nature
i	Entier

```
Bac 2009.py <
1 from numpy import*
2 def saisie_N():
3     test=True
4     while test:
5         N=int(input('Donner la taille du tableau'))
6         if 6<=N<=100:
7             test=False
8     return N
```

```
9 def saisie_D(N):
10    test=True
11    while test:
12        D=int(input('Donner un entier'))
13        if N % D==0 and D>1:
14            test=False
15    return D
```

```
16 def remplir(T,N):
17     for i in range(N):
18         test=True
19         while test:
20             T[i]=input('T['+str(i)+']=')
21             if 'a'<=T[i]<='z':
22                 test=False
```

```

23 def trier (T,N,D,M):
24     deb=0
25     fin=D
26     for i in range(M):
27         for j in range(deb,fin-1):
28             posmin=j
29             for k in range(j+1,fin):
30                 if T[k]<=T[posmin]:
31                     posmin=k
32             vide=T[j]
33             T[j]=T[posmin]
34             T[posmin]=vide
35             deb=fin
36             fin=fin+D
37 def Affichage (T,N):
38     for i in range (N):
39         print(T[i],end=' ')

```

```

40 N=saisie_N()
41 D=saisie_D(N)
42 M=N//D
43 T=array([str]*N)
44 remplir(T,N)
45 trier(T,N,D,M)
46 Affichage(T,N)

```

Exercice d'application : Bac théorique 2013

Problème : (12 points)

Soit T un tableau de N entiers (avec $6 \leq N \leq 50$). On se propose de trier le tableau T dans l'ordre croissant en utilisant le principe suivant :

1. On parcourt le tableau T de gauche à droite en comparant les éléments de T deux à deux ($T[i]$ avec $T[i+1]$) et en les permutant si nécessaire. Le premier parcours permet de placer le plus grand élément dans la dernière case.
2. On parcourt le tableau de droite à gauche (sans tenir compte de la dernière case : case triée) tout en comparant chaque deux éléments consécutifs de T et en les permutant si nécessaire pour placer le plus petit élément à sa bonne place (case n°1).
3. On refait les étapes 1 et 2 en parcourant le tableau tantôt de gauche à droite et tantôt de droite à gauche sans tenir compte des cases triées. Le traitement sera arrêté lorsque le tableau est trié.

Exemple

Soient $N=6$ et le tableau T suivant :

13	6	4	20	5	9
1	2	3	4	5	6

En appliquant le principe décrit ci-dessus sur le tableau T précédent, on obtient les étapes suivantes :

1. Le 1^{er} parcours de gauche à droite, permet de placer la valeur 20 (plus grand élément de T) dans la case n°6.

6	4	13	5	9	20
1	2	3	4	5	6

2. Le 1^{er} parcours de droite à gauche (sans tenir compte de $T[6]$), permet de placer la valeur 4 (plus petit élément de T) dans la case n°1.

4	6	5	13	9	20
1	2	3	4	5	6

3. Le 2nd parcours de gauche à droite (sans tenir compte de $T[1]$ et $T[6]$) permet de placer la valeur 13 (plus grand élément de la portion du tableau allant de la case 2 à la case 5) dans la case n°5.

4	5	6	9	13	20
1	2	3	4	5	6

4. Durant le 2nd parcours de droite à gauche (de la case 4 à la case 2), aucune permutation n'a été faite, donc le tableau est trié.

4	5	6	9	13	20
1	2	3	4	5	6

On se propose d'écrire un programme qui permet de remplir un tableau T par N entiers puis de trier T selon le principe décrit précédemment et d'afficher le tableau trié.

Questions

1. Analyser le problème en le décomposant en modules.
2. Analyser les modules envisagés.

Correction de l'exercice : Bac théorique 2013

Programme principale :

Algorithme **tri_gauch_droit**

Début

Saisie(N)

Remplir(T, N)

Trier(T, N)

Afficher(T)

Fin

TDNT

Type
Tab = Tableau de 50 entiers

TDOG

Objet	Type
N	Entier
T	Tab
Saisie	Procédure
Remplir	Procédure
Trier	Procédure
Afficher	Procédure

Procédure Trier (@T : Tab ; n : entier)

Début

j←0

Répéter

Tri←Vrai

Pour i de j à n-2 faire

Si T[i]>T[i+1] alors

Vide←T[i]

T[i]←T[i+1]

T[i+1]←vide

Tri←faux

Fin si

Fin pour

Si (Tri = Faux) alors

Tri \leftarrow Vrai

Pour i de n-1 à (j+1) (pas=-1) faire

Si T[i] < T[i-1] alors

Vide \leftarrow T[i]

T[i] \leftarrow T[i-1]

T[i-1] \leftarrow vide

Tri \leftarrow faux

Fin si

Fin pour

Fin si

j \leftarrow j+1

n \leftarrow n-1

Jusqu'à (Tri = Vrai)

Fin

8. Chapitre 8 : Les algorithmes de recherche

8.1. Introduction

Un algorithme de recherche dans un tableau est un algorithme qui permet d'y rechercher un élément vérifiant des critères définis à l'avance.

8.1.1. Recherche séquentielle dans un tableau non traité

Algorithme

Fonction Rech_seq (T : tab ; n : entier ; x : entier) : Booléen

Début

$i \leftarrow 1$

Trouve \leftarrow Faux

Tant que $i < n$ et Trouve = Faux faire

Si $T[i] = x$ alors

Trouve \leftarrow Vrai

Fin si

$i \leftarrow i + 1$

Fin Tant que

Retourner Trouve

Fin

En python

```

1 def recherche_seq(T,n,v):
2     i=0
3     Trouve=False
4     while (i<n) and (Trouve==False):
5         if T[i]==v:
6             Trouve=True
7         i=i+1
8     return Trouve

```

8.1.2. Recherche dichotomique dans un tableau triée :

Algorithme :

Fonction rech_dicho (T : tab ; n : entier ; x : entier) : Booléen

Début

d \leftarrow 0

f \leftarrow n-1

Trouve \leftarrow Faux

Tant que (d < f) et (Trouve = Faux) faire

M \leftarrow (d+f) div 2

Si T[M] < x alors

d \leftarrow M+1

Sinon si T[M] > x alors

f \leftarrow M-1

Sinon

Trouve \leftarrow Vrai

Fin Si

Fin Tant que

Retourner Trouve

Fin

Python :

```
1 def recherche_dicho(T,n,v):
2     d=0
3     f=n-1
4     Trouve=False
5     while (d<f) and (Trouve==False):
6         m=(d+f)/2
7         if T[m]<v:
8             d=m+1
9         elif T[m]>v:
10            f=m-1
11        else:
12            Trouve=True
13
14 return Trouve
```

9. Chapitre 9 : Interface graphique

9.1. Introduction

Les interfaces graphique GUI sont des interface homme-machine, sont appelées GUI : Graphical User Interface. Elles permettent à l'utilisateur d'interagir avec un programme informatique, grâce aux différentes objets graphiques :

- Les zones de textes
- Les cases à cocher
- Bouton radio
- Bouton poussoir
- Menu

Ces objets graphiques sont généralement actionnés avec un dispositif de pointage, le plus souvent la souris

La création d'une GUI peut se réaliser par deux méthodes :

- Ecrire le code qui lui correspond en utilisant un langage de programmation.
- Créer les objets graphiques (cités ci-dessus) directement sur une fenêtre en utilisant un outil de conception et de création d'interfaces graphiques (un éditeur graphique tel que Qt Designer ou Qt Creator) qui contient un éditeur d'interfaces utilisateur Glisser et déplacer (Drag and Drop)

9.2. Qt Designer + PyQt5

Pour la préparation de l'environnement du travail

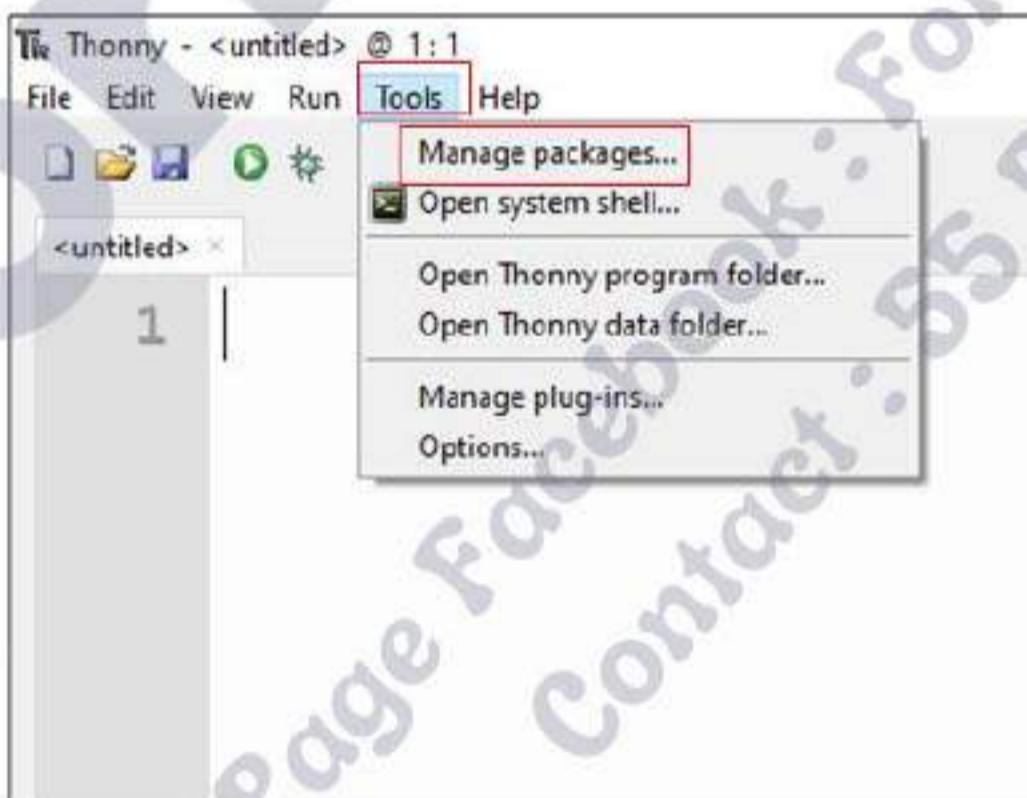
1^{ère} étape : La première étape pour commencer à créer des interfaces graphiques et les faire lier à votre programme, il faut télécharger et installer le logiciel Qt designer, qu'est un outil de conception et de création d'interfaces graphique via le lien suivant :

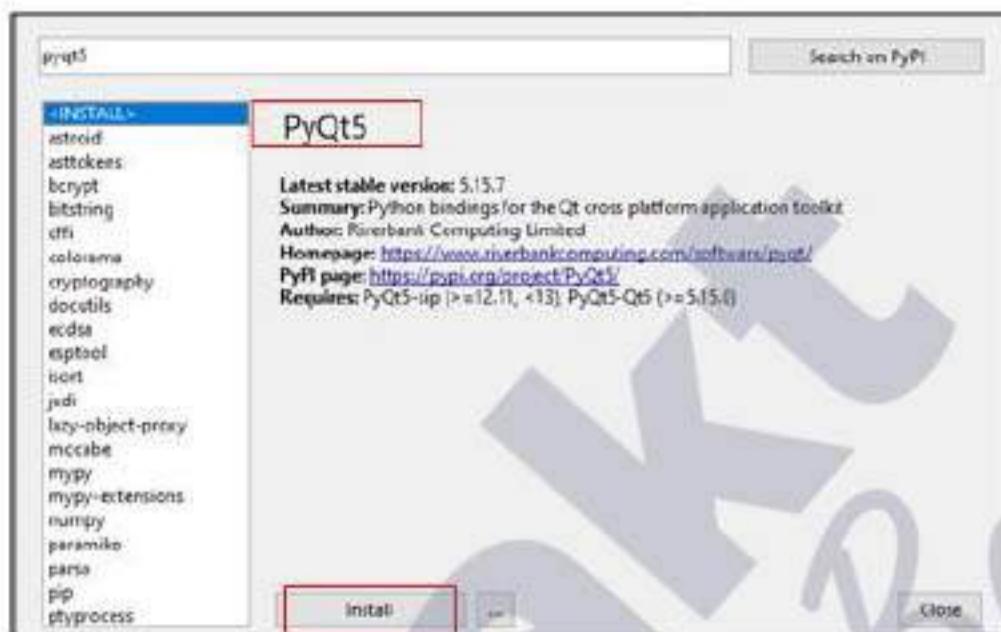
[Qt Designer Download for Windows and Mac \(fman.io\)](https://www.fman.io/qt-designer-download/)



2^{eme} étape : La deuxième étape consiste à installer les packages PyQt5. Ce package on peut l'installer avec l'invite commande Windows ou directement avec Thonny :

- Ouvrir Thonny/ Menu Tools/ Manage packages
- Taper le nom du package à installer (PyQt5)
- Cliquer sur install





PyQt5 est une bibliothèque qui permet de créer des interfaces graphiques avec Python.

Le module PyQt est considéré comme une liaison du langage Python avec la boîte outils GUI toolkit Qt.

PyQt implémente environ 440 classes et plus de 6000 fonctions et méthodes.

PyQt6 est la version utilisée actuellement par python.

Dans cette formation, et dans le cadre académique, on va intéresser à la version PyQt5.

PyQt5 offre trois méthodes de création d'interface graphique :

- * L'interface est programmée puis exécutée (fichier.py)
- * L'interface peut être créée par un outil de création d'interface graphique GUI (fichier.ui) puis converti en un code python (fichier .py) pour qu'il soit exécuté.
- * L'interface peut être créée par un outil de création d'interface graphique GUI (fichier.ui) puis appelé ce fichier dans le code python (fichier.py). Cette méthode est l'avantage majeur de ce module.

Pour créer une fenêtre graphique en PyQt5, on doit :

- ✓ Importer la classe qui génère l'application : QApplication depuis le package PyQt5.QtWidgets.
- ✓ Importer la classe QWidget depuis le package PyQt5.QtWidgets
- ✓ Créer une fenêtre avec la méthode :

- ✓ Visualiser la fenêtre à l'aide de la méthode
- ✓ Exécuter l'application à l'aide de la méthode :

Dans cette formation, et à cadre académique, on va intéresser à la troisième méthode de création d'interface graphique avec PyQt5. C'est-à-dire, on va créer l'interface par l'outil Qt Designer puis on va appeler le fichier.ui dans le code python (fichier.py) crée par l'éditeur Thonny.

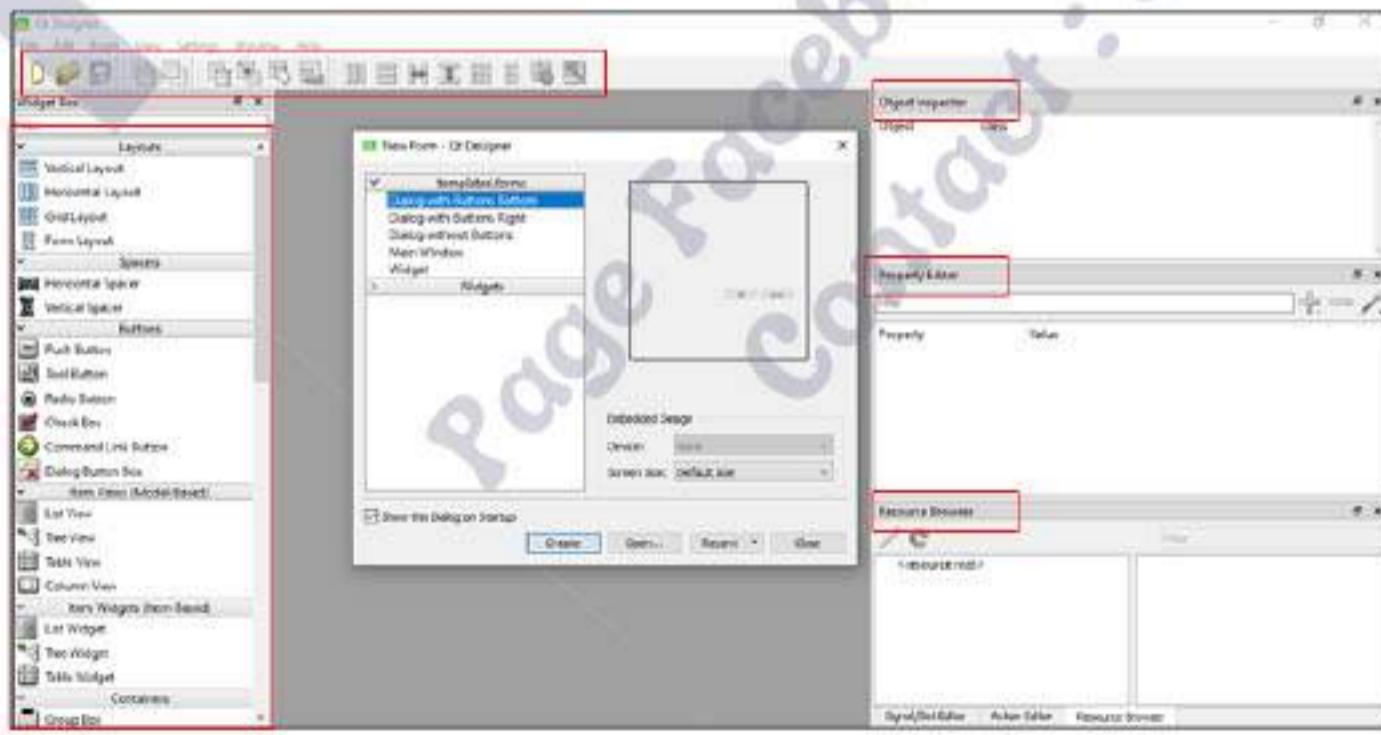
9.3. Utilisation de l'outil Qt5 designer

Qt Designer est un générateur d'interface graphique, elle permet de créer vos fenêtres visuellement, elle vous permet de créer des objets graphiques (widgets) et modifier ses propriétés (Nom, Taille, couleur, ...) d'utiliser des lay-outs, etc.

Le widget est un composant d'interface graphique (Label, zone de texte, bouton radio, case à cocher, bouton poussoir, menu, liste, ...)

La fenêtre principale de Qt Designer fournit une barre de menus (Menu bar) et fournit également une barre d'outils (toolbar') qui affiche les options couramment utilisées.

Elle comprend également d'autres composantes qui fournissent un ensemble de fonctionnalités et d'outils.



On peut modifier les propriétés du fenêtre principales à partir de l'éditeur de propriétés comme exemple :

- ☒ Changer le nom de l'objet (Window)
- ☒ Ajouter un titre pour la fenêtre
- ☒ Changer la taille de la fenêtre.
- ☒ Changer la taille de la fenêtre (largeur et hauteur)
- ☒ Changer le fond et la couleur
- ☒ Ajouter une icône pour la fenêtre

On peut ajouter des widgets (Label, line Edit, Push Button, Radio Button, Check Box, Combo Box, List Widget, ...) directement sur une fenêtre avec le mécanisme glisser et déposer (drag and drop) à partir de la boîte de widgets.

9.4. La partie programmation

Le fichier crée par Qt Designer possède l'extension .ui (User Interface)

Après avoir réalisé l'interface graphique de votre application, on doit passer à la partie programmation. Pour lancer un éditeur Python et écrire le code correspondant à votre application.

☒ Remarque : De préférence, il faut mettre les deux fichiers (fichier.ui) et fichier.py dans le même répertoire.

Pour commencer on doit :

- ❖ Importer la bibliothèque PyQt5 et les modules pour réaliser les interfaces graphiques

```
from PyQt5 import QtWidgets, uic
```

- ❖ Importer la classe qui permet de créer l'application :

```
Nom_Objet_App = QApplication()
```

- ❖ Charger le fichier .ui crée par Qt Designer

```
Nom_objet_fen = uic.loadUi("Nom_fichier.ui")
```

- ❖ Visualiser la fenêtre à l'aide de la méthode show()

Nom_objet_fen.show()

- ❖ Générer un signal (événement) : Exemple clicked : Appel de la fonction

Nom_objet_fen.NomWidget.clicked.connect(Nom_module)

- ❖ Exécuter l'application à l'aide de la méthode exec_0 :

Nom_objet_App.exec_0

- ❖ Créer le sot : Le module qui va être appelée en réponse d'un signal.

9.5. Exemple pratique

Nous allons créer une application de calcul arithmétique qui calcul le Plus Grand Diviseur Commun de deux entiers M et N strictement positifs avec une interface graphique et en utilisant la méthode de différence.

La méthode de différence :

PGCD (6,18) = PGCD (6,12) = PGCD (6,6) =6

PGCD (27,63) = PGCD (27,36) = PGCD (

Etape n°1 : Conception de l'interface

Notre interface, dans la plupart des exercices et sujets sera composés seulement par :

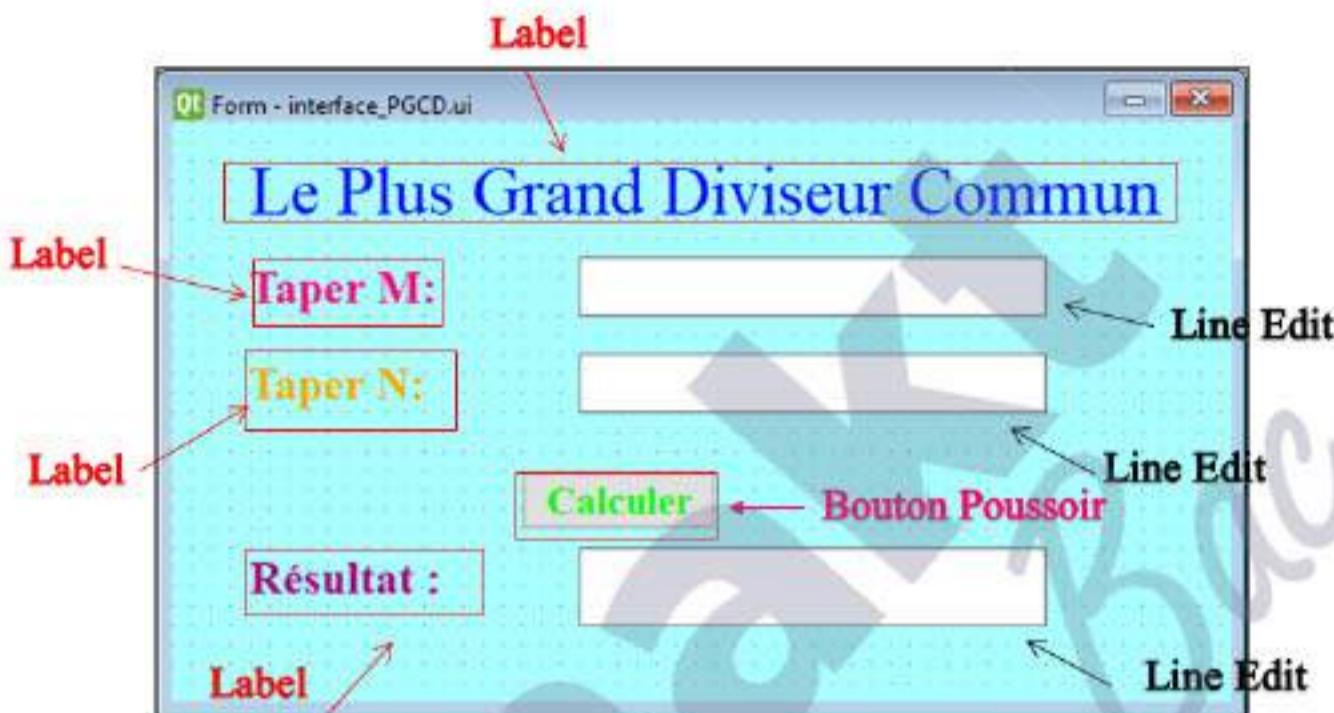
Des labels : Les zones pour écrire et afficher des messages et titres.

Des Line Edit : Sont les zones de saisie, les champs dans lesquels l'utilisateur va saisir les données nécessaires.

Des boutons poussoirs : Qui seront utiliser ensuite pour exécuter l'application.

Ces widgets sont disponibles dans le tool bar à gauche, il suffit de les glisser et les déposer dans les places souhaitées.

Remarque : On peut afficher le résultat final sur un label vide ou dans une line edit vide.



On peut la remplacer par un
label vide

Une étape très importante dans la phase de conception de l'interface graphique est la définition des noms des label, lines edit et les boutons à appeler ou à utiliser dans le code python.

Sont généralement les zones de saisi des données à traiter par le programme, le bouton et le label ou line edit consacrée à l'affichage du résultat du programme.

Pour définir le nom, il suffit de cliquer sur le Widget souhaité puis aller aux propriétés d'objet dans menu bar à droite comme indiqué dans la figure suivante :



Après avoir terminé la préparation de l'interface, il faut l'enregistrer dans un dossier bien déterminé avec un nom bien déterminé.

En ce qui concerne le développement du code python, nous avons déjà corrigé l'exercice du PGCD dans le chapitre 4 de ce livre.

```

PGCD.py
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 def meth_diff(M,N):
4     test=True
5     while test:
6         if M>N:
7             M=M-N
8         elif M<N:
9             N=N-M
10        if M==N:
11            test=False
12    return M
13
14 def PGCD():
15     M_ch=windows.M.text()
16     N_ch=windows.N.text()
17     M=int(M_ch)
18     N=int(N_ch)
19     if not (M>0):
20         msg='Veuillez vérifier la valeur de M SVP!'
21     elif not (N>0):
22         msg='Veuillez vérifier la valeur de N SVP!'
23     else:
24         x=meth_diff(M,N)
25         msg='PGCD ('+M_ch+', '+N_ch+') = '+str(x)
26         windows.res.setText(msg)

```

Importer la bibliothèque PyQt5 et les modules pour réaliser les interfaces graphiques

La fonction `meth_diff` : c'est fonction qui prend en paramètre deux entiers M et N et retourne le PGCD par la méthode de différences (Voir chapitre 4)

C'est le module qui s'exécute suite à une clique sur le bouton.

Le module qui s'exécute suit à une clique sur le bouton poussoir, doit être toujours une procédure qui ne retourne pas de résultat. Son rôle est d'appeler la fonction qui calcul le PGCD et affiche le résultat dans le label/ mine edit consacrée pour l'affichage du résultat.

Ce module sert aussi à récupérer les données saisis par l'utilisateur dans les lines edit.

Une remarque très importante à ce stade, c'est que les données sont toujours récupérées sous la forme des chaînes de caractères. On doit donc les convertir en entiers pour qu'on puisse les utiliser dans la fonction `meth_diff`.

Mais, comment on récupère les valeurs à partir les lines edit ?

C'est avec la commande suivante :

`Nom_objet_fen.Nom_line_edit.text()`

Comment on affiche le résultat donc ?

C'est avec la commande suivante :

`Nom_objet_fen.Nom_line_edit.setText(le message à afficher)`

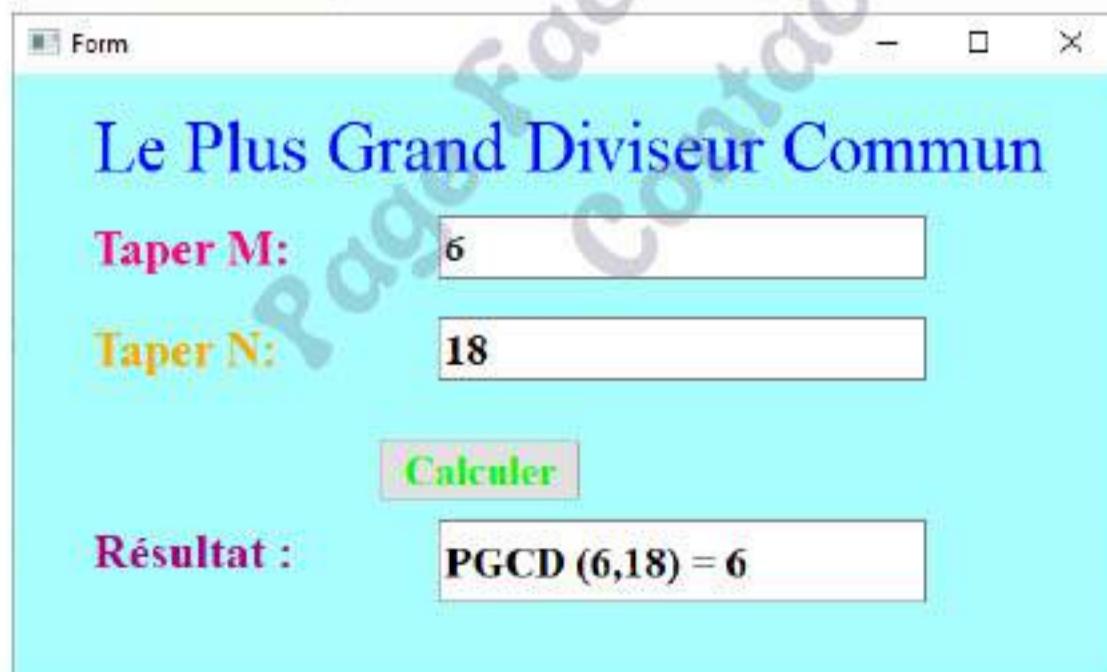
Enfin dans le programme principal du notre code, on doit :

- * Charger le fichier .ui créé par Qt Designer
- * Visualiser la fenêtre à l'aide de la méthode `show()`
- * Générer l'événement : La clique sur le bouton "Calculer" et le faire connecter au module PGCD

```
app=QApplication([])
windows=loadUi("interface_PGCD.ui")
windows.show()
windows.calculer.clicked.connect(PGCD)
app.exec()
```

Le code est enregistré dans le même dossier que le fichier .ui.

Maintenant, lorsque on exécute le code via l'éditeur Thonny, l'interface que nous avons créée s'ouvre automatiquement.



Important :

- ☒ Dans la partie graphique, nous n'utilisons ni la méthode input () pour saisir des données, ni la méthode print () pour afficher des résultats ou des messages.
- ☒ Dans la partie graphique, nous n'utilisons pas les structures Répéter et tant que pour contrôler les données saisis par l'utilisateur. Il ne s'agit pas d'un traitement répétitif, si l'utilisateur a saisi une valeur fausse un message d'erreur s'affiche dans le label/ line edit du résultat. Il doit effacer l'ancienne valeur et re-saisir une autre valeur.
- ☒ Vous pouvez ajouter un bouton clear afin de vider les zones de saisie.