REPUBLIQUE TUNISIENNE
MINISTERE DE L'EDUCATION
Direction régionale de l'enseignement
Ben Arous

Cours Algorithme et Programmation 4 ème Sc/Math

Auteur

Kaouther Allegui
Enseignante informatique
Lycée 9 avril Boumhal

Sommaire

Partie 1 : Démarche de résolution de problèmes

I. Introduction :

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

II. Les étapes de résolution d'un problème

Activité 1 :

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

a. Spécifier les différentes données nécessaires pour résoudre ce problème
b. Proposer le traitement nécessaire pour avoir le résultat voulu à partir les données
c. Indiquer le résultat à obtenir.
d. Ecrire un algorithme permettant de résoudre ce problème.
e. Implémenter cet algorithme en Python.

Solution :

· Données :

.....
.....

· Traitement :

.....
.....

· Résultat :

.....
.....

Algorithme	Implémentation en Python

Algorithme .....
 Début
 .....
 .....
 .....
 .....
 .....
 .....
 .....
 Fin
	.....
	 .....
	 .....
	 .....
	 .....
	
	 Résultat d'exécution
	 `Taper Lo :2.5`
	 `Taper La :1.25`
	 `La surface = 3.125 Le périmètre = 7.5`

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

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

Partie 2 : Les structures simples

I. L'opération d'entrée :

1. Définition :

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

2.Vocabulaire et syntaxe :

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

Exemples

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

3. Remarque :

♣ Quand on demande à la machine de lire une variable, cela implique que l'utilisateur va devoir écrire cette valeur. ♣ Les données qui sont lues doit être compatibles aux variables réservées en mémoire.

II. L'opération de sortie :

1. Définition :

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

2. Exemple :

En Algorithme	En Python
Affichage d'un texte : Ecrire ("La valeur de n est : ")	print ("La valeur de n est : ")
Affichage de contenu d'une variable : Ecrire (n)	print (n)
Affichage mixte (texte et variable) : Ecrire ("La valeur de n est : ", n)	print ("La valeur de n est : ", n)

Remarque :

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

4. Application :

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

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

En déduire le Tableau de Déclaration des Objets (TDO)

Objet	Type/Nature
…… …… ……	…… …… ……

III. L'opération d'affectation :

1.Définition :

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

2.Vocabulaire et syntaxe :

En Algorithme	En Python
Variable□ valeur	Variable = valeur

3.Exemple :

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

Remarque :

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

Application 1 :

Soit la séquence d'affectations suivante :

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

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

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

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

....

3. Quel est le rôle de cette séquence ?

....

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

....

Application 2 :

1. Compléter le tableau suivant :

Instruction	Valeur de A	Valeur de B
A□5		
B□7		
A□A+B		

B□A-B		
A□A-B		

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

....

3.Application 3 :

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

Solution :

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

T.D.O :

Objet	Type/Nature
…….	……
……	……
…….	……

Partie 3 : Les structures de données

I. Les constantes et les variables :

1. Les constantes :

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

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

2.Déclaration :

Tableau de Déclaration des Objets :

Objet	Type/Nature
Nom_const	Constante de valeur

Exemples :

pi=3,14

Tableau de Déclaration des Objets :

Objet	Type/Nature
pi	Constante de valeur 3,14

2. Les variables :

Une variable est un objet dont la valeur est susceptible de changer au cours de l'exécution d'un programme.
Une variable est caractérisée par :

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

1.Activité 1 :

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

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

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

II. Les types de données :

1. Le type Entier (int) :

a. Définition :

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

b. Déclaration :

Objet	Type/Nature

Nom_variable	entier

Exemple :

Objet	Type/Nature
a	entier

c. Les opérations arithmétiques et relationnelles sur les entiers :

Opérations (en Algorithme)	Opérations (en Python)	Rôles	Exemples
+	+	Adition	5 + 2 = 7
-	-	Soustraction	5 – 2 = 3
*	*	Multiplication	5 * 2 = 10
/	/	Division	5 / 2 = 2.5
=	= =	Egalité	5 = = 5 renvoie vrai
<	<	Inférieur	2 < 5 renvoie vrai
>	>	Supérieur	5 > 2 renvoie vrai
≤	<=	Inférieur ou égal	2 <= 5 renvoie vrai
≥	>=	Supérieur ou égal	5 >= 2 renvoie vrai
≠	!=	Différent	5 != 2 renvoie vrai
n ∈ [5,20]	5<=n<=20	Appartient	5<=3<=20 renvoie vrai
X^n	X ** n	Puissance	5 ** 2 = 25
MOD	%	Modulo	5 % 2 = 1
DIV	//	Division entière	5 // 2 = 2

Application :

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

Solution :

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

2. Le type Réel (float) :

a. Définition :

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

b. Déclaration :

Objet	Type/Nature
Nom_variable	entier

Exemple :

Objet	Type/Nature
x	réel

c. Les opérations arithmétiques et relationnelles sur les réels :
Les mêmes opérations que les entiers sauf DIV et MOD.
d. Les fonctions prédéfinies sur les réels :

Les fonctions arithmétiques standards relatives aux types entier et réel :

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

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

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

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

Objet	Type/Nature
……	……

3. Le type Booléen (bool) :

a. Définition :

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

b. Déclaration :

Objet	Type/Nature
Nom_variable	booléen

Exemple :

Objet	Type/Nature
b	booléen

c. Opérations logiques sur les Booléens :

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

La table de vérité qui manipule ces opérateurs est :

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

Application :

Compléter le tableau suivant :

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

4. Le type Caractère (str):

a. Définition :

Un caractère (chiffre où lettre où symbole) est représenté le caractère lui-même mis entre guillemets
Exemple : "A", "a", "+",...

Remarque :

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

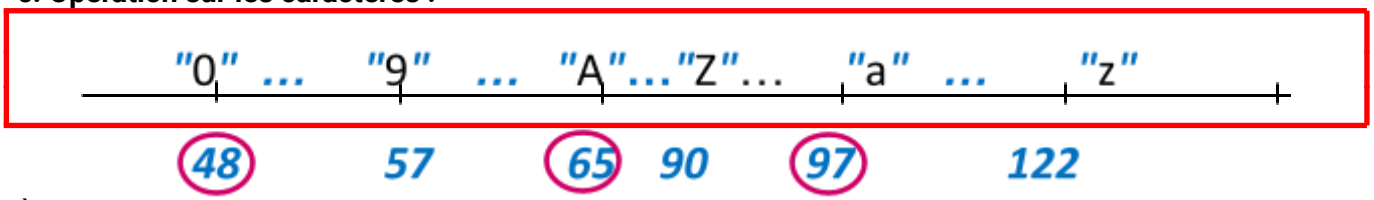
b. Déclaration :

Objet	Type/Nature
Nom_variable	caractere

Exemple :

Objet	Type/Nature
c	caractere

c. Opération sur les caractères :



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

Les opérations usuels : +, =, , <=, >=, <>.
Exemple :
"A" < "B" est une proposition vraie car 65<66.
"a" >"b" est une proposition fausse car 97 <98

d. Fonctions prédéfinis sur les caractères :

Nom en algorithme	Code en Python	Rôle	Exemples
n□ord (c)	n=ord(c)	Retourne le code ASCII du caractère c	print(ord('A'))donne ……. print(ord('a')) donne ……

c□chr(n)	c=chr(n)	Retourne le caractère dont le code ASCII est n	print(chr(65)) donne .....

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

5. Le type chaîne de caractère (str):
a.Définition : C'est une succession de n caractère (lettre, symbole, chiffre) avec n ≥ 0.
Une chaîne de caractère doit être placée entre deux guillemets ou entre apostrophes en algorithme et en python
Remarque :
♣ Si n = 0 alors la chaîne est dite vide ("" : chaîne vide).
♣ Les valeurs chaîne de caractères sont définies entre guillemets
b. Déclaration :

Objet	Type/Nature
Nom_variable	Chaine de caractère

Exemple :

Objet	Type/Nature
ch	Chaine de caractère

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

Exemple
Ch=

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

Remarque : en python les chaînes de caractères sont immuables ((c-à-d **non modifiable** : ni **changement**, ni **suppression** et ni **insertion ni tri**.).
Il est donc, interdit d'écrire par exemple : ch[0]='P'

d. Opérateurs

Opération	Algorithme	Python	Exemple

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

e. Fonctions prédéfinis sur les chaînes de caractère :

Algorithme	Python	Description	Exemple
l□long(ch)	**l=len(ch)**	Retourne le nombre de caractères de la chaine ch.	ch□"2020" l□long(ch) l contient ……
p□pos (ch1, ch2)	**p=ch2.find(ch1)**	Retourne la première position de ch1 dans ch2. Si ch1 n'existe pas dans ch2, retourne -1	ch1□"2" ch2□"2020" ch3□"3" p□pos(ch1,ch2) p contient …… p□pos(ch3,ch2) p contient ……
ch□convch(X)	**str(X)**	Retourne la conversion un nombre X en chaine.	ch□convch (2020) ch contient ……
a□estnum(ch)	**ch.isdigit()**	Retourne Vrai si la chaine ch est convertible en numérique, Faux dans le cas contraire.	a□estnum("25") a contient …… b□estnum("a25") a contient ……
n□valeur(ch)	n = int(ch)	Retourne la conversion d'une chaine ch en numérique entière si c'est possible.	ch□"2020" n□valeur (ch) n contient …… ch□"2Info2" n□valeur (ch) ……
	n= float(ch)	Retourne la conversion d'une chaine ch en numérique réelle si c'est possible.	ch□"12.8" n□valeur (ch) n contient ……
ch1□sous_Chaine (ch, d, f)	**ch1=ch[d:f]**	Retourne une partie de la chaine ch à partir de la position d jusqu'à la position f exclue.	ch1□souschaine("baccalauréat 2023",5,12) Ch1 contient ……
ch1□effacer (ch, d, f)	**ch1=ch[:D]+ch[F:]**	Efface des caractères de la chaine ch à partir de la position d jusqu'à la position f exclue.	ch□"baccalaureat" ch1□effacer(ch, 3,12) ch1 contient ……

ch1□majus(ch)	**ch1=ch.upper()**	Convertit la chaine ch en majuscule.	ch1□majus(”Devoir”) ch1contient ……

Serie1

Exercice n°1 Donner l'affichage adéquat pour chaque instruction après l'exécution :

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

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

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

Exercice n°2 Donner le résultat de chacune des instructions suivantes :

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

Exercice n°4 : Soit Ch="DevoirdeSynthese" Déterminer la valeur de chaque objet

Actions	Resultat
x=len(ch) print(x)	
y=ch[-1] print(y)	
z=ch[:6] print(z)	
a=ch.isdigit() print(a)	
b=ch.upper()	

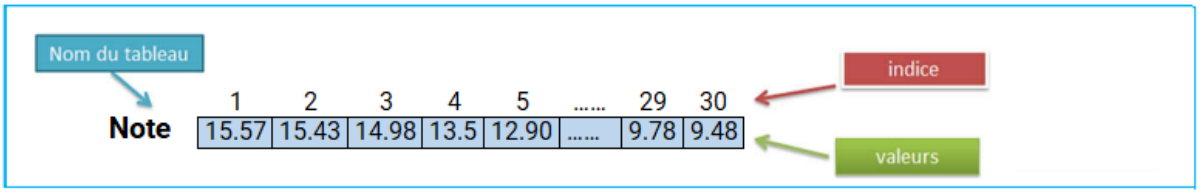
print(b)	
f=ch.find("e") print(f)	

6.Le tableau à une dimension (Vecteur) :

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

Un tableau est caractérisé par :

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



b. Déclaration d'une variable de type vecteur :

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

Syntaxe :

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

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

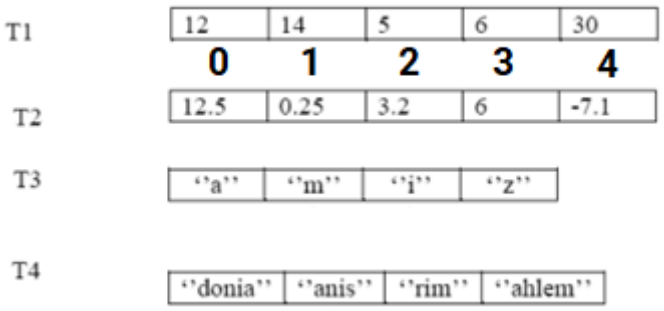
En Algorithme	En Python
Tableau de déclaration des objets **Objet** : note — **Nature / Type** : Tableau de 30 réels	from numpy import array note =array ([float ( )] * 30)

Remarque :

♣ Un vecteur est une suite de cases mémoires qui peut contenir des valeurs de même type.

♣ Un vecteur est caractérisé par son nom, sa taille et les types de ses éléments.

Application 1: Déclarer les 4 tableaux suivants en algorithme et en python



Correction :

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

c. Accès aux éléments d'un vecteur :

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

Pour accéder en lecture ou en écriture au $i^{ième}$ élément d'un tableau, il suffit de spécifier l'identificateur du tableau suivi de l'indice i entre deux crochets

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

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

T[0] : permet de renvoyer la valeur 12

T[3] : permet de renvoyer la valeur 66

T[8] : permet de renvoyer la valeur 9

Exemple2 : Soit T un tableau de 10 réels

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

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

d. Modifier les éléments d'un vecteur :

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

Syntaxe :

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

Remarque :

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

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

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

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

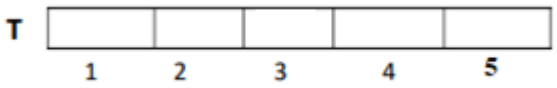
0	1	2	3	4	5	6	7	8	9
99	76	1	66	55	0	4	8	9	10

¬ **Activité :** Soit T un vecteur de 5 entiers

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

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

Solution :



e.Affichage du contenu d'un tableau à une dimension :

L'instruction d'écriture "print()" s'applique aux tableaux comme aux variables plus une boucle répétitive "for …"

Syntaxe :

Pour afficher le contenu d'un tableau

On utilise la structure Pour afin de parcourir le tableau.

//Afficher les éléments du tableau un par un

En Algorithme	En Python

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

Exemple :

Soit T un tableau de 10 valeurs d'entières.

Pour afficher le tableau T :

En Algorithme	En Python
…… ……	…… ……

……	

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

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

Syntaxe :

//remplir les éléments du tableau un par un

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

Exemple :

Soit T un tableau de 10 valeurs d'entières.

Pour saisir le tableau T :

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

Exercice d'application 2 :

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

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

Remarque :

♣ On a fait deux boucles successives pour plus de lisibilité, mais on aurait tout aussi bien pu n'en écrire qu'une seule dans laquelle on aurait tout fait d'un seul coup.

Serie2

Exercice 1 :
Soit la séquence suivante :
V [1] □ 20
V [2] □ 3
V [3] □ V [1] DIV V [2]
V [4] □ V [3]*V [2]
V [5] □ V [1] + V[3]+V[4]

1) Déclarer le tableau V.
2) Quel est le contenu de chaque élément du tableau V ?

Exercice 2 :
Ecrire un programme qui saisit un tableau T de n lettres et affiche l'ordre alphabétique de chaque lettre.
On dispose de l'exemple suivant :

T	A	h	C	w	q

□Le programme affichera : "1-8-3-23-17"

*Exercice **3** :*
Soit T un tableau de n réels. On se propose d'écrire un programme intitulé OCCURRENCE qui saisit n et T puis affiche le nombre d'occurrences (nombre d'apparition) d'un réel x donné.
Exemple : pour x=6, le nombre d'occurrence=4

T	3.5	**6**	**6**	3	**6**	-1	10	0	2.5	**6**	-5

Partie 4 : Les structures de contrôle conditionnelles

Définition
La structure de contrôle conditionnelle permet à un programme de choisir et modifier son traitement selon une condition.
On distingue 3 formes de structures conditionnelles :

- Simple
- Généralisées
- À choix multiples

I. La structure de contrôle conditionnelle simple :

Activité 1 :
Ecrire un programme qui permet de saisir un entier n et d'afficher leur parité (paire ou impaire)
Exemples :
♣ 25 est impaire
♣ 36 est paire
D'après sur ce qu'on a déjà vu, on ne peut pas résoudre ce type de problème car on a amené à décider si l'entier n est paire ou impaire, donc on a intérêt de définir une nouvelle structure qui permet de résoudre ce problème.

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

b. Vocabulaire et syntaxe :

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

Remarque :

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

♣ La condition peut être simple ou composée.

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

✔ Lorsque la valeur de la condition est :

- VRAI : le Traitement 1 sera exécuté
- FAUX : le Traitement2 sera exécuté.

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

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

♣ Si traitement 2 est vide, on parle de structure conditionnelle simple réduite qui a la syntaxe suivante :

En Algorithme	En Python
Si condition alors <traitement1> Fin Si	if condition : <traitement1>

2.Solution de l'activité 1 :

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

Activité 2 : Ecrire un algorithme et son script python qui permet de saisir un entier « N » **strictement positif** et d'afficher sa racine carré.

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

Activité 3 : Ecrire un algorithme et son script python qui permet de saisir un entier « N » et d'afficher sa valeur absolue sans utiliser la fonction abs.

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

Activité N°4:Ecrire un algorithme intitulé **Existence** puis un script python qui permet de déterminer si une chaîne ch1 existe dans une chaîne ch2.

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

II. La structure de contrôle généralisée :

Activité 2 :

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

± Exemples :

♣ 5 est positif

♣ -9 est négatif

♣ 0 est nul Remarquez bien qu'on a plus que deux traitements donc on a intérêt de définir une nouvelle forme de structure de contrôle conditionnelle qui nous permet de résoudre ce type de problèmes

a. Définition :

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

b. Vocabulaire et syntaxe :

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

Remarque :

♣ L'exécution d'un traitement entraîne automatiquement la non-exécution des autres traitements.

♣ La condition peut être simple ou composée.

Solution de l'activité 2 :

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

Objet	Type/Nature
…….	…….
……	……

Activité 3 :

Ecrire un algorithme qui permet de saisir le numéro du mois puis afficher la saison correspondante

Exemple1 : mois=2 saison =hiver

Exemple2 : mois=5 saison =printemps

....

Solution de l'Activité 3

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

III. Structure de choix :

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

a. Définition :

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

b. Vocabulaire et syntaxe :

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

Remarque :

- Cette structure offre une meilleure lisibilité de la solution que la forme généralisée.
- Dans tous les cas **Sélecteur** doit être de type **scalaire** (pas un réel ou chaîne de caractère)
- En cas **d'égalité du sélecteur** à une valeur (ou fait partie d'une **liste**) l'action qui lui fait associer est exécutée. **Les autres ne seront pas exécutées**

- En cas **d'inégalités** il y aura exécution de la partie **Sinon** (Action n+1) si elle est présente (optionnelle)
- Chaque bloc peut comporter une ou plusieurs instructions.
- La clause sinon et son traitement sont facultatifs dans la structure.

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

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

Objet	Type/Nature
…… ……	…… ……

Application :

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

Solution de l'Application

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

Objet	Type/Nature
……	……

Serie3

Exercice1 :

Cocher □ la bonne réponse.

❶	x = int(input("x= ")) R=x **if** x<0 : R=-x print("|" + str(x)+ "| =" + str(R))	X=-9 Le programme affiche : ⬜ |-9| =-9 ⬜ |-9| =9 ⬜	9| =9

❷	chaine1= input("Chaîne 1 : ")
	chaine2= input("Chaîne 2 : ")
	
	if (**len**(chaine2) > **len**(chaine1))**:**
	  print (chaine2)
	else:
	  print (chaine1)
		Chaîne 1 : 2 Ti2
		Chaîne 2 : 2 Info2
		Le programme affiche :
		⬜ 2 Ti2
		⬜ 2 Info2
		⬜ chaine2

❸	A□ 7
	Si (A ≠ 7) alors
	  C□A*A
	Sinon
	  C□-A
	Finsi
		Après l'exécution du code suivant, quelle sera la valeur de C
		⬜ 7
		⬜ -7
		⬜ 49

❹	Quelle est la valeur affichée par la séquence d'instructions suivante : **A← 5** **B←10** **C← 15** **Si (A>B) et (A>C) alors** **Afficher(A)** **Sinon Si B >C alors** **Afficher(B)** **Sinon** **Afficher(C)** **Fin Si**	⬜ 5 et 15 ⬜ 5 ⬜ 15 ⬜ 10

Exercice2 :

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

Exercice3 :

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

Exercice4 :

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

Exemple 1: si ch = ″ technologie ″ et c =″ e ″
alors le prog affichera « e existe dans technologie″
Exemple 2 : si ch = ″ technologie ″ et c =″ r″
alors le prog affichera « r n'existe pas dans technologie″
Exercice5 :
Ecrire un algorithme puis un script qui permet de saisir successivement un réel (r1), un opérateur (op) et un 2ème réel (r2) puis d'afficher le résultat (R) de l'expression « r1 op r2 » si l'opération est possible SINON affiche les messages convenables en cas d'erreur.
Exemple1 : Pour r1= 50 , op= ″+″ , r2=20 le programme affichera 50.0 +20.0 = 70.00
Exemple2 : Pour r1= 5 , op= ″*″ , r2= 9 le programme affichera 5.0 * 9.0 = 45.00

Partie 5 : Les structures contrôle itératives

Une structure répétitive est aussi appelée boucle nous permettent de gagner énormément de temps en éliminant les répétions. Imaginez que vous avez à exécuter une certaine instruction un certain nombre de fois, disons 100 fois, alors au lieu de taper ces instructions 100 fois, les structures répétitives nous permettent de la taper une seule fois en indiquant le nombre de fois que l'ordinateur doit l'exécuter.
On distingue deux types de boucles :
- Une structure itérative complète où le nombre d'itérations est connu à l'avance : La boucle **Pour … Faire …**
- Une structure itérative à condition d'arrêt où le nombre d'itérations est inconnu à l'avance.
    - La boucle **Répéter … Jusqu'à …**
    - La boucle **Tant que … Faire …**

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

Activité 1 :
Ecrire un programme qui permet d'afficher le mot « Informatique » 1000 fois.
Φ Le nombre de répétition de l'instruction [Ecrire (" Informatique ")] est très grand ce qu'il est impossible d'écrire ces nombres d'instructions donc on a intérêt de définir une nouvelle structure appelé la structure de contrôle itérative complète que nous permet de répéter l'exécution d'une instruction un nombre connu de fois.

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

b.Vocabulaire et syntaxe :

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

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

Activité 2 :
Ecrire un programme en python dans lequel on affiche la table de multiplication d'un entier donnée

Solution de l'activité 2 :

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

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

Application :

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

……

……

Exercice 3 :

Algorithme	Python
Algorithme …… **Début** …… …… …… …… …… **Fin**	…… …… …… Ou bien …… …… ……

Objet	Type/Nature
……	……
……	……

T.D.O :

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

II. La structure itérative à condition d'arrêt : La boucle [Répéter … Jusqu'à] :
Activité 2 :
Ecrire un algorithme qui demande à l'utilisateur un nombre N compris entre 1 et 3 jusqu'à ce que la réponse convienne. Demander de ressaisir si sa réponse incorrecte.

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

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

Remarque :
· La condition d'arrêt est considérée comme une condition de sortie de la boucle car, une fois elle est vérifiée on quitte la boucle.
· La condition de sortie de la boucle peut être simple ou composée.
· La boucle Répéter … Jusqu'à est une structure adaptée pour le contrôle de la saisie des données. Elle impose l'utilisateur d'entrer des données qui vérifient certaines contraintes.
· S'il y a un éventuel compteur, il faut l'initialiser avant la boucle pour assurer son avancement dans la boucle.
Correction Activité 2 :

En Algorithme	En Python

Algorithme controle
Début
……
……
……
……
Fin

……
……
……

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

Objet	Type/Nature
n	entier

Application : Soit l'algorithme suivant :

Algorithme

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

Exécuter manuellement cet algorithme pour les valeurs suivantes :

¬ n=23

n	C	D	S

¬ n=46

n	C	D	S

Déduire le rôle de cet algorithme :

……

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

Activité 3 :

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

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

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

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

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

b. Définition :

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

c. Vocabulaire et syntaxe :

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

4. Remarque :

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

Solution de l'activité 3 :

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

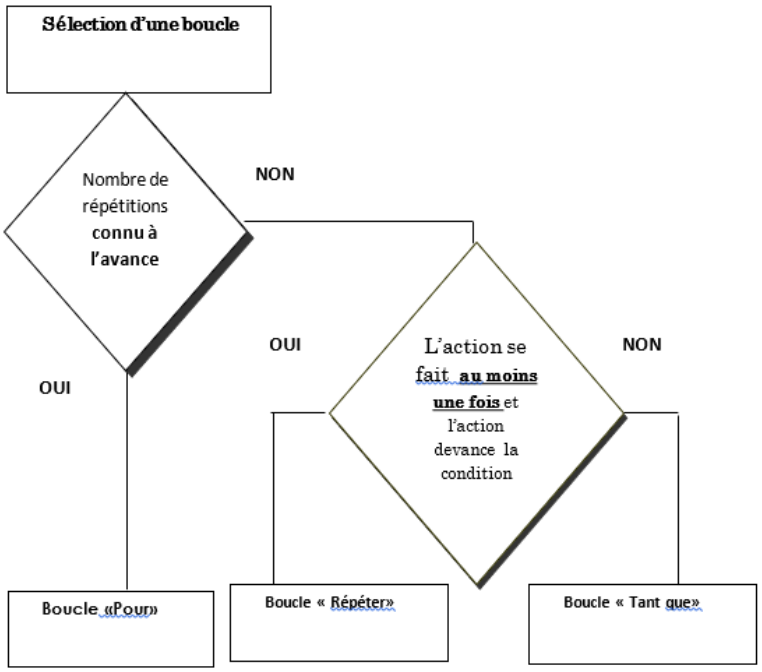
Remarques :

La structure **while** réalise un nombre d'itérations inconnu.

le flux d'exécution pour une instruction while :

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

SELECTION D'UNE BOUCLE répétitive



Serie4

Exercice1 ☺ Cocher ☐ la bonne réponse.

❶	x = int(input("x= ")) y= int (input("y= ")) s=1 **for i in range(y+1)** : s= s +x print(s)	X=3 et y= 5 Le programme affiche : ☐ 18 ☐ 19 ☐ 14
❷	x = int(input("x= ")) y= int (input("y= ")) s=0 **for i in range(x,y+1)** : s= s +i print(s)	X=3 et y= 5 Le programme affiche : ☐ 10 ☐ 11 ☐ 12
❸	x=0 y=1 for i in range(1,5): x=x+i y=y*i print('x=',x,'y=',y)	Le programme affiche : ☐ x= 20 y=25 ☐ x= 10 y=24 ☐ x= 10 y=40
❹	X = 5 Y = 0 for i in range(1,4): X= X+1 Y= Y+1	

4- Soit la séquence d'instructions suivantes en python :

```
for i in range(0, 4):
    print(i)
```

Quelle est la série de nombre affichée par ce programme ?

☐ 0, 1, 2 et 3
☐ 0, 1, 2, 3 et 4
☐ 0, 1, 2, 3, 4 et 5
☐ 1, 2, 3 et 4

Exercice 2 :
Ecrire un programme qui calcule la somme des N premiers termes positifs.
Exercice 3 :
Ecrire un programme qui calcule la somme des N premiers termes positifs impaires.
Exercice 4 :
Ecrire un programme qui calcule la somme des N premiers termes positifs pairs non multiple de 3.
Exercice 5 :
Ecrire un programme python intitulé **PARFAIT** qui permet de déterminer si un nombre entier supérieur à 1 est parfait. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs.
Exercice 6 :

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

Exercice n° 7:
Ecrire un algorithme qui permet de lire un mot M qui ne dépasse par 15 caractères

Exercice n° 8:
Ecrire un algorithme qui permet de lire un entier de 3 chiffres

Exercice n° 9:
Ecrire un algorithme qui permet de lire un entier de pair

Exercice n°10 :
Ecrire un algorithme intitulé **PALINDROME,** qui permet de lire un mot M qui ne dépasse par 15 caractères et de vérifier s'il est palindrome ou non**. (Un mot palindrome se lit dans les deux sens tel que les mots RADAR, ELLE, …)**.

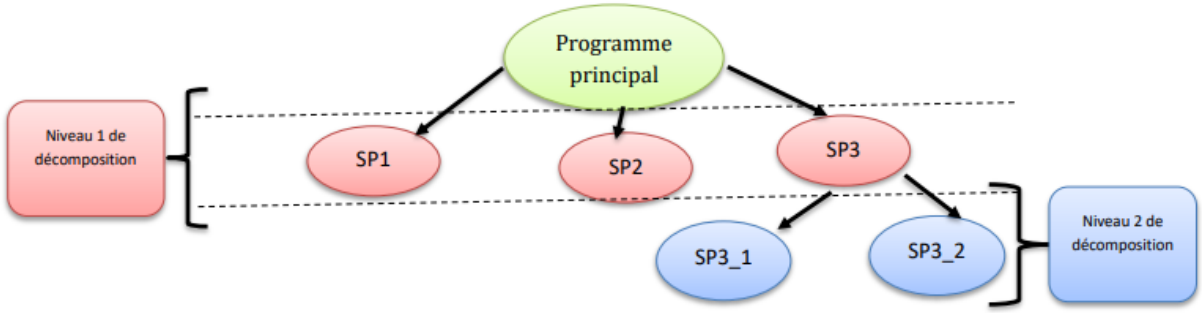
Exercice n°11 :
Ecrire un programme qui permet de saisir une suite d'entier et de calculer leur somme à chaque saisir et afficher cette somme. La liste d'entier se termine par la valeur **-1**.
***Exemple* :** Si la suite saisie est 5, 6, 7, -4, -1 la valeur affichée est 14.

Exercice n°12 :
Ecrire un algorithme qui permet de saisir un entier N de 3 chiffres et vérifier s'il est cubique ou non.
Exemple : $153=1^3+5^3+3^3$ est un nombre cubique.

Partie 6 : Les sous programmes

Introduction :

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



--Schéma 1 : *décomposition modulaire*

I. Les fonctions :

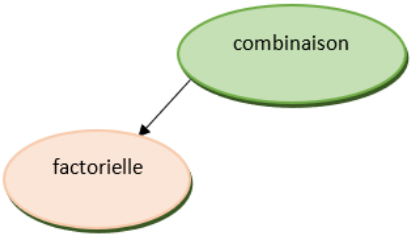
Activité 1 :

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

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

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

Décomposition :

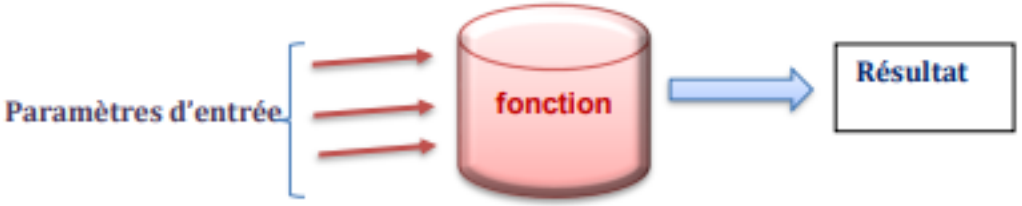


Remarque :

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

a. Définition :

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



- Une fonction qui ne retourne pas de résultat est une procédure

b. Déclaration :

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

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

Solution de l'activité 1 :

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

Remarques :

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

c.Appel d'une fonction :

Une fonction peut être appelée à partir:

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

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

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

	* **Tant que** (nom_fonction(parE1,..., parEn) = valeur	* **while**(nom_fonction(parE1,...,parEn) == valeur)

RETENONS:

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

Solution de l'activité 1 :

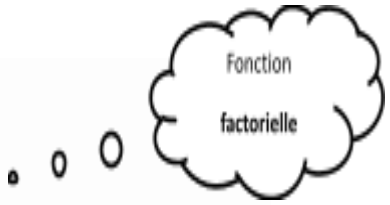
En algorithme	En python
Algorithme du programme principal **Algorithme ……** **Début** …… …… …… …… …… **Fin**	…… …… …… ……

T.D.O globaux

Objet	Type
……	……
……	……
……	……

En python :

```
1  def factorielle (x) :
2    f=1
3    for i in range(1,x+1):
4            f=f*i
5    return f
6  n=int(input('donner n'))
7  p=int(input('donner p'))
8  r=factorielle(n)/(factorielle(p)*factorielle(n-p))
9  print(r)
```

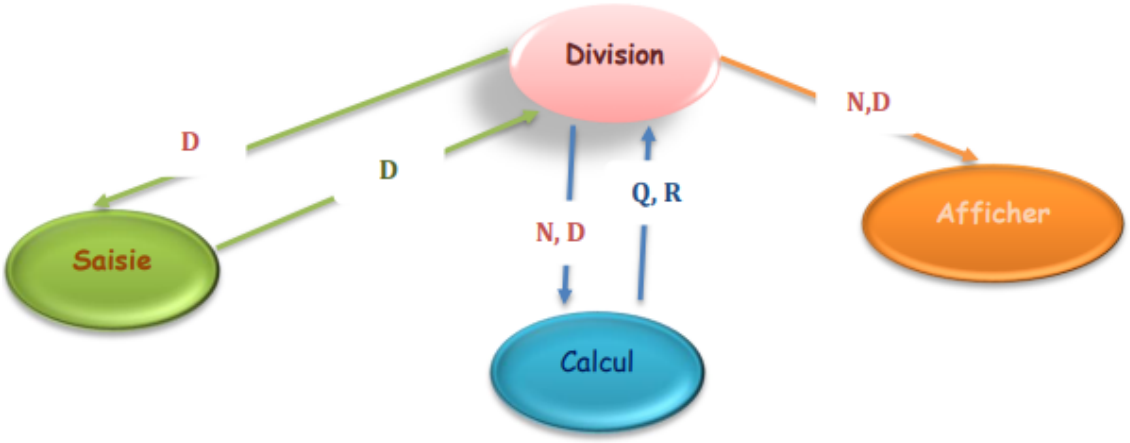


II. Les procédures :

Activité n°2 :

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

Décomposer le problème en module.

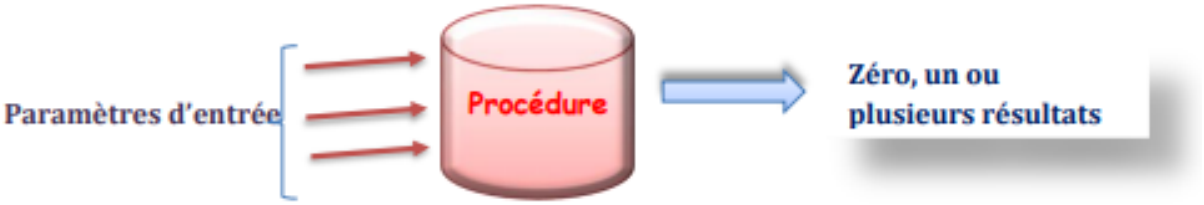


Pour résoudre ce problème on peut utiliser :

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

a. Définition :

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



b. Déclaration :

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

Remarque :

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

Solution de l'activité 2 :

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

T.D.O locaux

Objet	Type
…… ……	…… ……

c.Appel d'une fonction

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

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

Solution :

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

T.D.O globaux

Objet	Type
…… ……	…… ……

Remarque :

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

A. Paramètres formels et effectifs :

Activité n°1 :

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

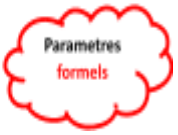
Solution :

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

TDO Globaux :

Objet	Type
x,y,m	réel
max	fonction

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



TDO Locaux :

Objet	Type
c	réel

En python : solution 1

……
……
……
……
……
……
……
……
……
……

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

En python : solution2

……
……
……
……
……
……

……
……
……
…….

a. Les types de paramètres :
Il existe deux types de paramètres :

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

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

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

b.Portée des variables :
Selon l'emplacement de déclaration, on peut avoir deux types de variables (objets) dans une fonction : des variables locales ou des variables globales.
La portée désigne l'emplacement de définition et la durée de vie d'une variable (objet).

- **Variable locale :**

Elle est déclarée dans le corps d'un sous-programme. Elle n'est accessible qu'à l'intérieur de module dans lequel a été déclarée.
Exemple : **c** de la fonction **Max**

Exemple 1:

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

Résultat d'exécution

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

Dans ce X est un variable locale
```

Exemple 2:

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

Résultat d'exécution

```
La valeur de X est : 2
X=4
X est un variable globale
```

- **Variable globale :**

Elle est définie en dehors d'un sous-programme. Elle est visible et utilisable dans tout le programme mais la fonction ne peut pas la modifier.
Exemple **: x, y et m** du programme **maximum**

Remarque : en python il est possible de modifier une variable globale dans un module si seulement si cette variable est déclarée dans le sous-programme avec le mot global.
Voir solution 1 programme division

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

- **Le passage par valeur :**

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

- **Le passage par référence :**

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

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

Serie5

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

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

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

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

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

Exemple : Pour **N=6** :

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

Les mots totalogramme sont : temporairement, ses, elle

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

Exemple :

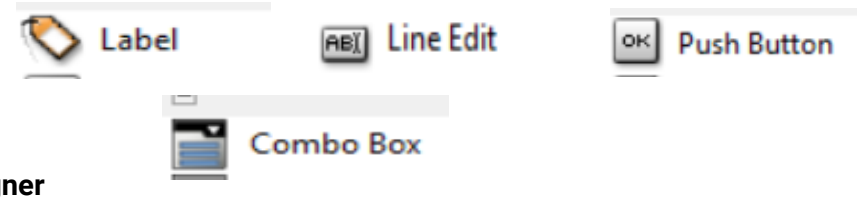
3	7	4	9	8

Données d'entrée Saisir le nombre d'éléments : n=5 T
Données de sortie Maximum = 9

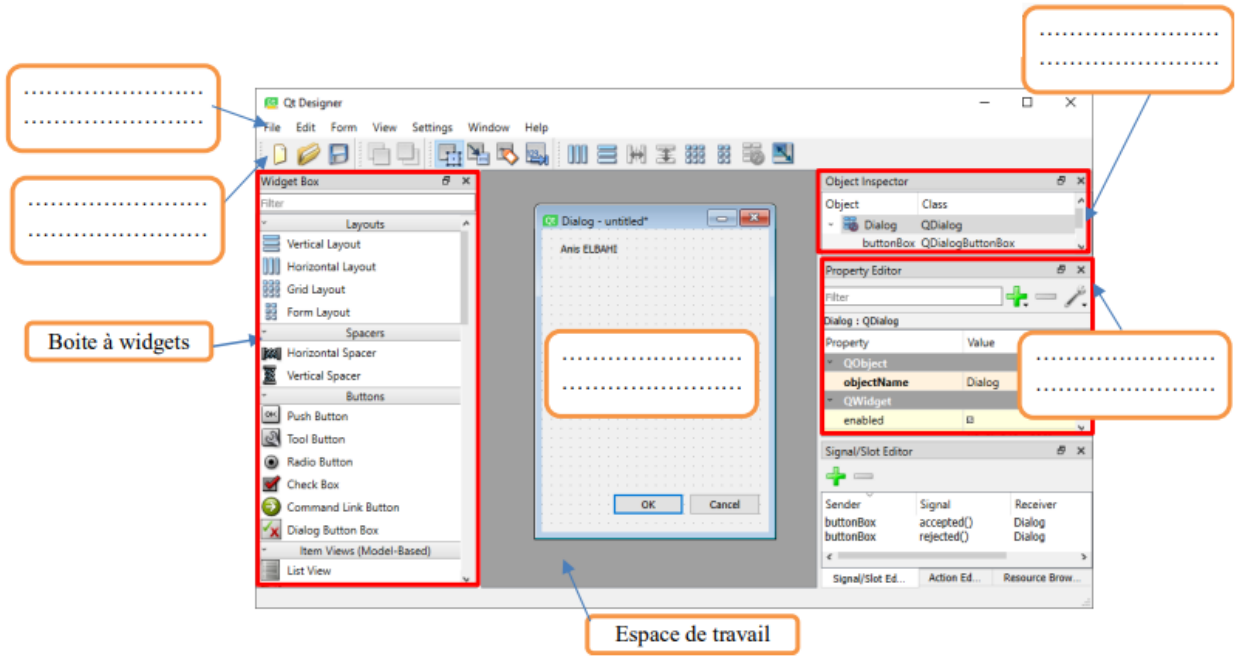
Les Interfaces graphiques

Introduction :
Les interfaces graphiques (ou interfaces homme/machine) sont appelées GUI (Graphical User Interface). Elles permettent à l'utilisateur d'interagir avec un programme informatique, grâce aux différents objets graphiques (zone de texte, case à cocher, bouton radio, bouton poussoir, menu, …).
Ces objets graphiques sont généralement actionnés avec un dispositif de pointage, le plus souvent la souris.
Dans le présent Cours on utilisera **Qt Designer** pour créer des interfaces Graphiques et Python pour les programmer

Qt Designer est un logiciel qui permet de créer des interfaces graphiques. Il est basé sur la technique de **glisser-déposer** en fournissant des éléments prédéfinis appelés **widgets** (gadgets pour Windows).



Présentation de Qt Designer
Compléter par le nom de chaque rubrique.



Pour créer une application Qt graphique (GUI), il faut passer par 2 etapes
<u>Etape n°1:</u>
Concevoir l'interface graphique du projet à l'aide de l'outil QtDesigner en utilisant la technique « Glisser– Déposer »
<u>Etape n°2:</u>
A l'aide du langage de programmation python programmer les widgets insérés dans l'interface en apportant les bibliothèques nécessaires.

TP1

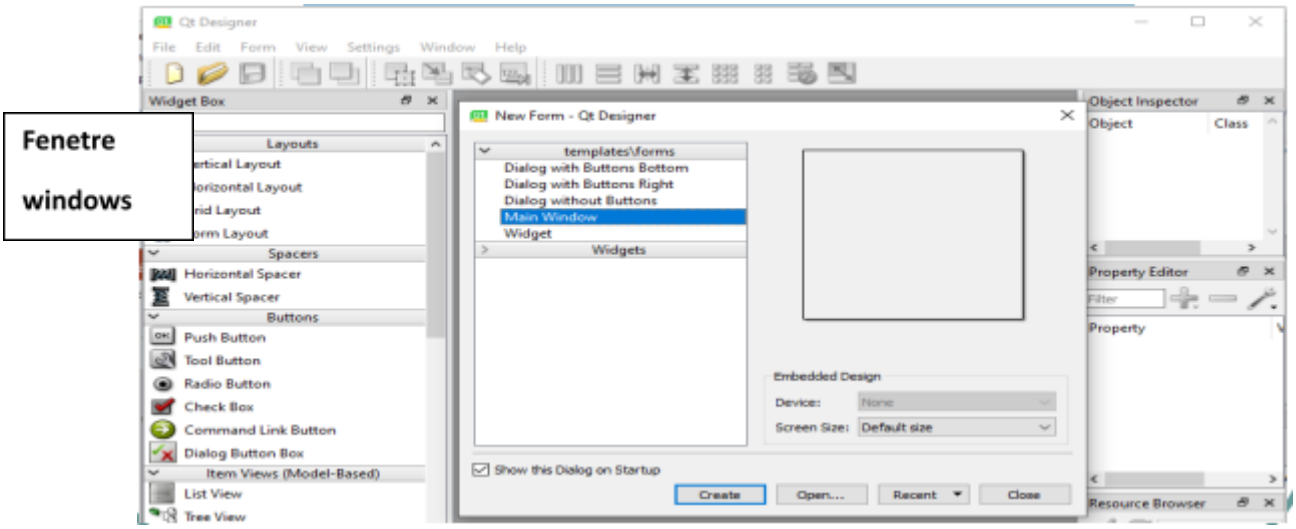
Créer un dossier portant le nom **tp1** dans **C:/bac2023** qui va contenir les deux fichiers : **somme.ui** et **calcul.py**

1.Réaliser une interface graphique (sous **QtDesigner**) contenant deux entrés **a** et **b**, et un résultat qui va contenir la somme de a et b.

a-Lancer **QtDesigner** et créer l'interface graphique suivante et et l'enregistrer sous le nom : **somme.ui** dans votre dossier de travail

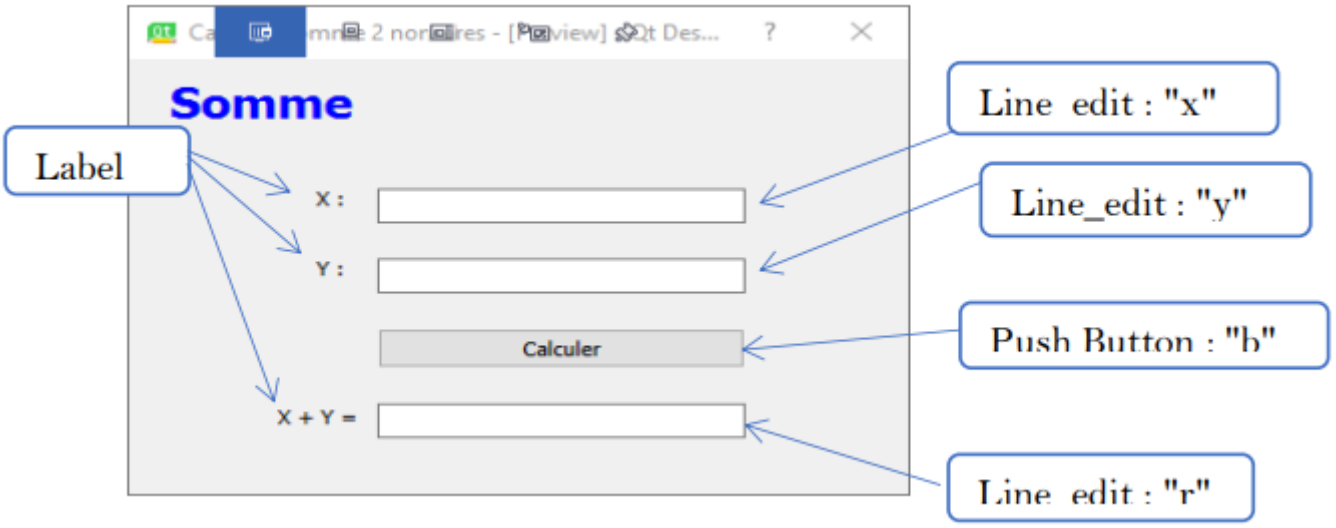
Démarrer Qt Designer

Pour créer une fenêtre principale pour une application : Menu File / New / Main Window / Create

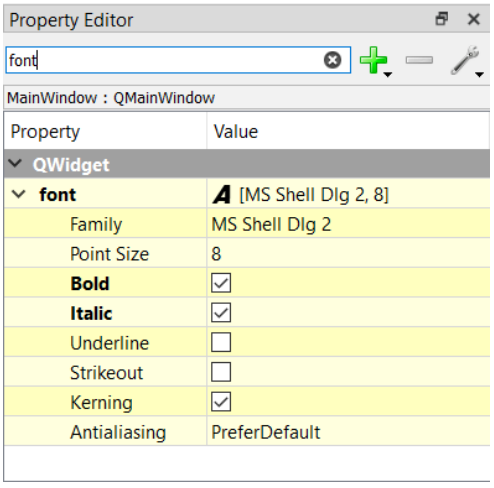


b.Insertion d'un objet:

Le clic sur le bouton calculer permet de calculer et d'afficher la somme de x et y dans la zone correspondante.

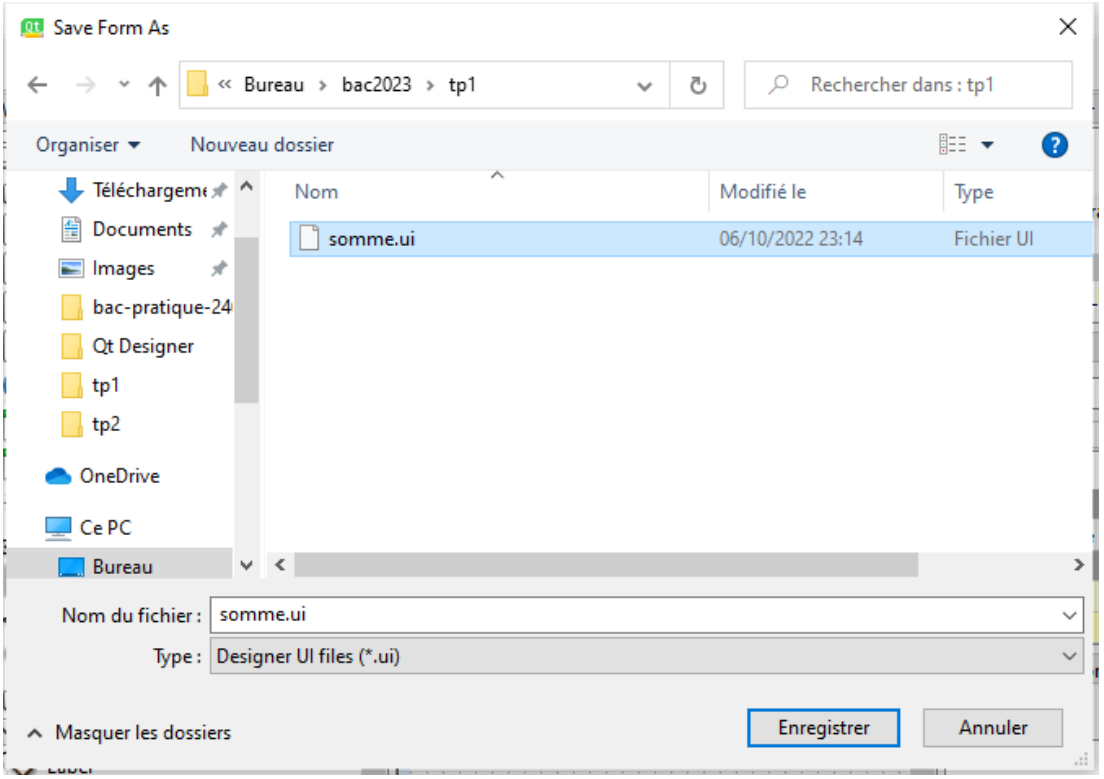


c.Modifier les propriétés des objets (taille ,couleur, police, couleur arrière plan, objectname….)



d.Nommer la fenêtre

e. Enregistrer l'interface sous le nom somme.ui

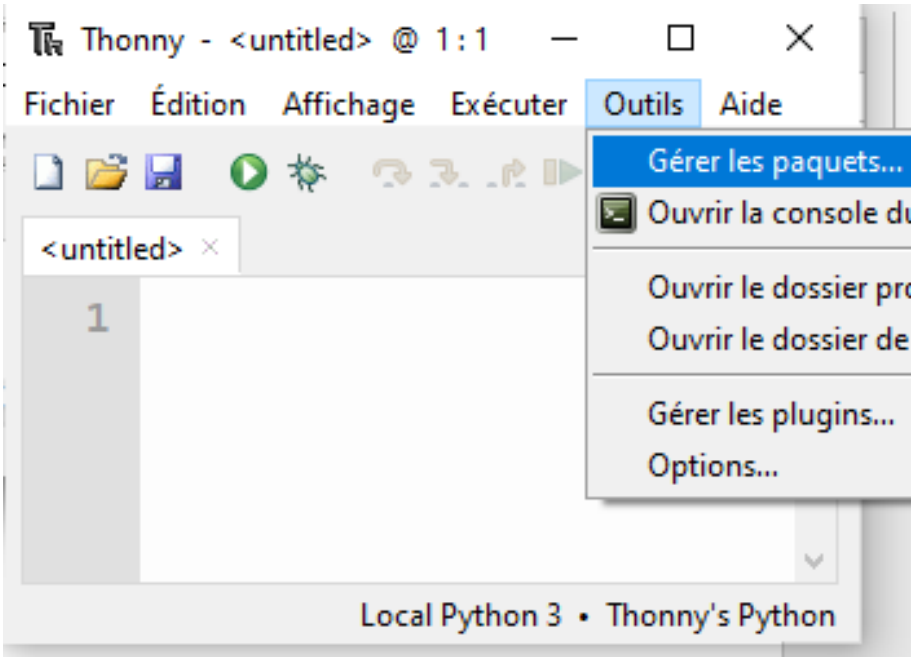


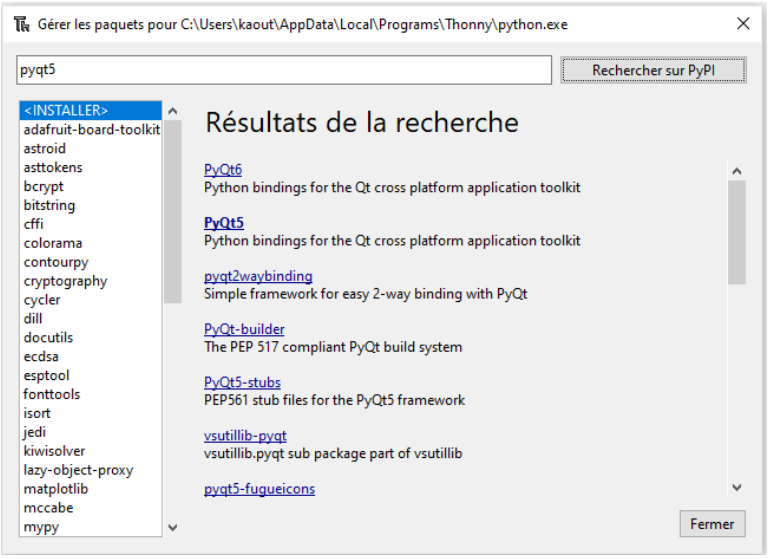
2.Programmation des objets
a.Lancer l'éditeur Thonny après avoir l'installer.

b.Installer le module PYQT5

Ouvrir thonny / Menu tools / Manage packages

♣ Taper le nom du package à installer (PyQt5, pyqt5-tools ou bien PyQt5Designer)





♣ Cliquer sur install

PyQt est un module qui permet de lier le langage Python avec la bibliothèque Qt , Il permet ainsi de créer des interfaces graphiques en Python.

c. Ecrire le format général

Annexe

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
.....
.....
app = QApplication([])
windows = loadUi ("Nom_Interface.ui")
windows.show()
windows.Nom_Bouton.clicked.connect (Nom_Module)
app.exec_()
```

d.Implémenter la fonction somme au dessus du programme principal

e.Activer le bouton calculer en faisant appel à la fonction somme

windows.b.clicked.connect(somme)

Nom_bouton =b

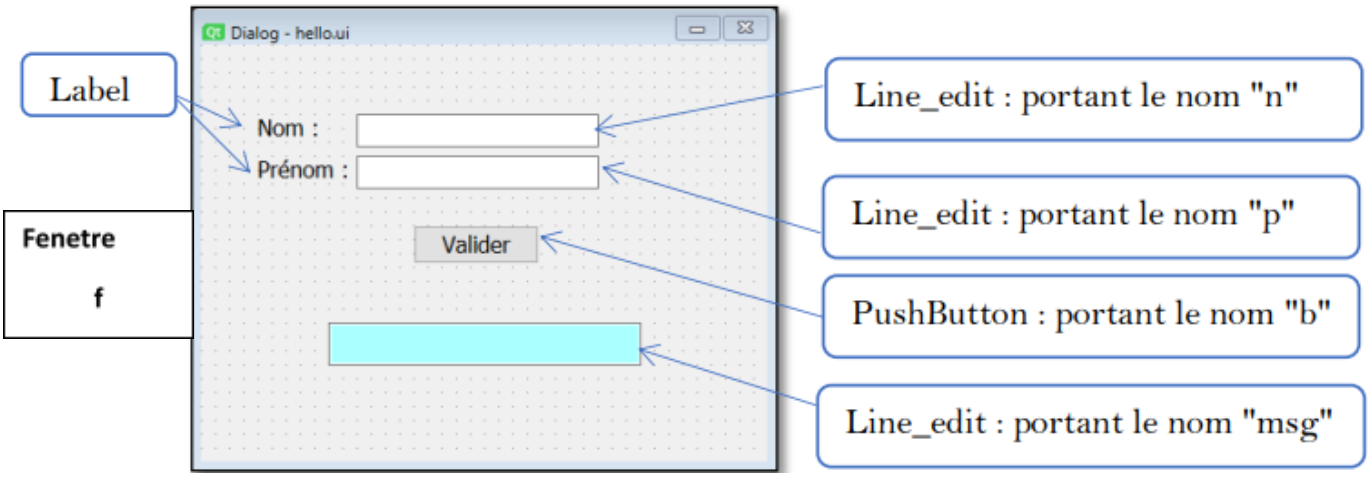
f.Enregistrer sous le nom : calcul.py dans le dossier **tp1** du **C:/bac2023**

```
1  from  PyQt5.uic  import loadUi
2  from  PyQt5.QtWidgets import QApplication
3  #les fonction
4  ……
5  ……
6  ……
7  ……
8  ……
9  ……
10 ……
11 ……
12 #pp
13 app = QApplication([])
14 windows = loadUi (" …… ")
15 windows.show()
16 #bouton calculer
17 ……
18 #execution de l'application
19 app.exec_()
```

TP2

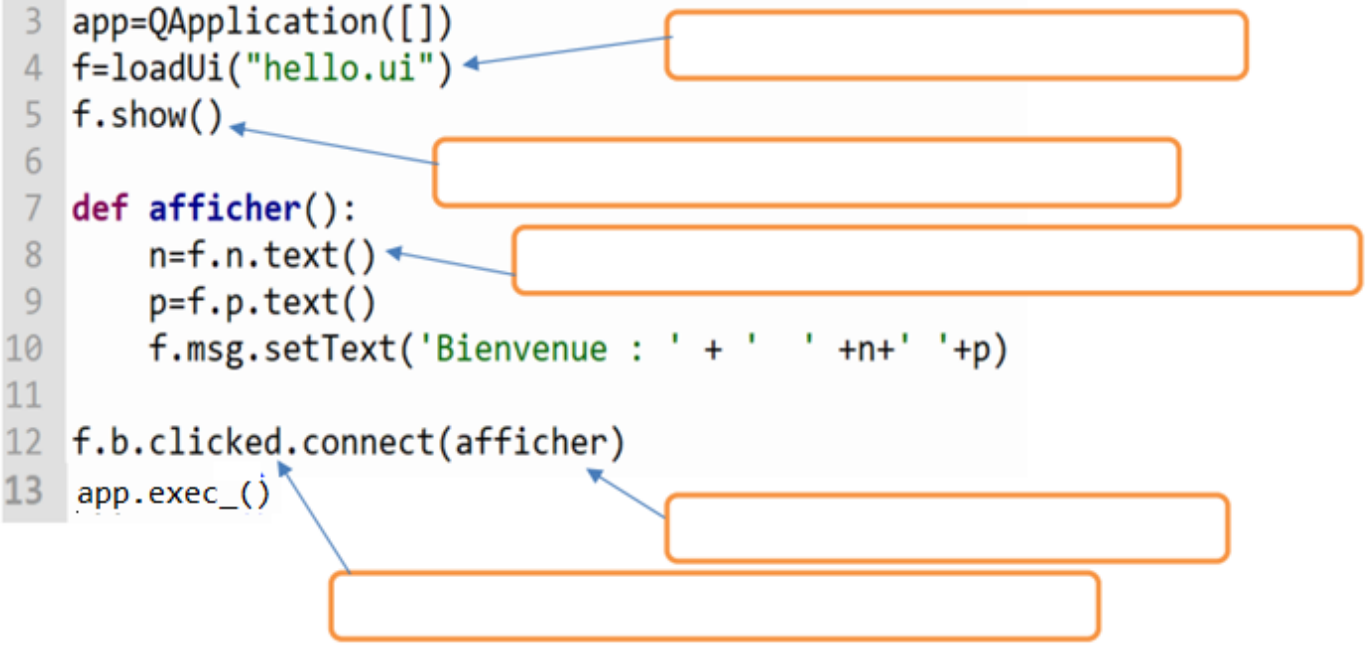
Enregistrer les fichiers résultats dans un même dossier appelé « tp2 » dans C:/bac2023

1-Lancer **QtDesigner** et créer l'interface graphique suivante et enregistrer-la sous le nom "**hello**.ui" :



2- Lancer Thonny, puis écrire le programme (et l'enregistrer sous le nom : afficher.py dans votre dossier de travail) qui Après avoir remplir les zones nom et prénom et à la suite du clic sur le bouton valider , le message «Bienvenue nom prénom" sera affiché dans la zone « msg » comme le montre l'interface ci-contre .

```
1  from PyQt5.uic import loadUi
2  from PyQt5.QtWidgets import QApplication
3  app=QApplication([])
4  f=loadUi("hello.ui")
5  f.show()
6
7  def afficher():
8      n=f.n.text()
9      p=f.p.text()
10     f.msg.setText('Bienvenue : ' + '   ' +n+' '+p)
11
12 f.b.clicked.connect(afficher)
13 app.exec_()
```

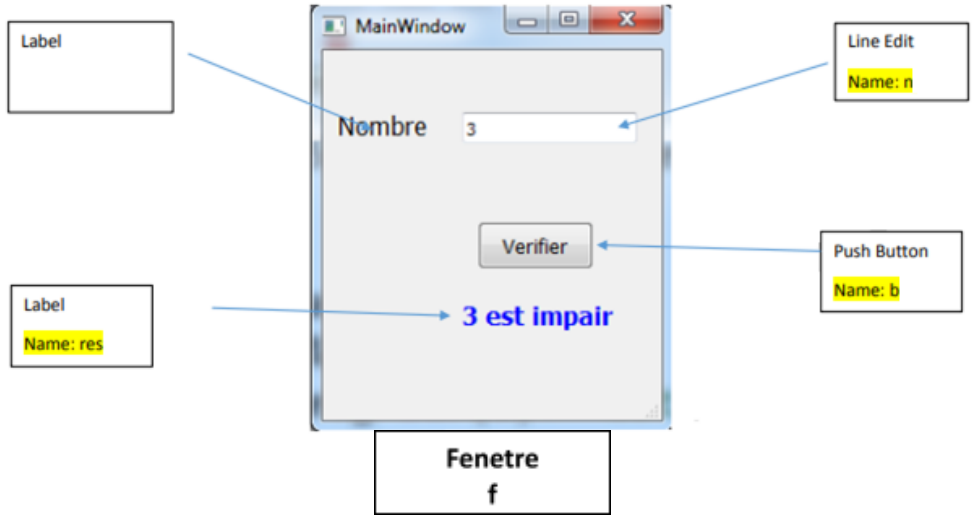


TP3

Enregistrer les fichiers résultats dans un même dossier appelé « tp3 » dans C:/bac2023

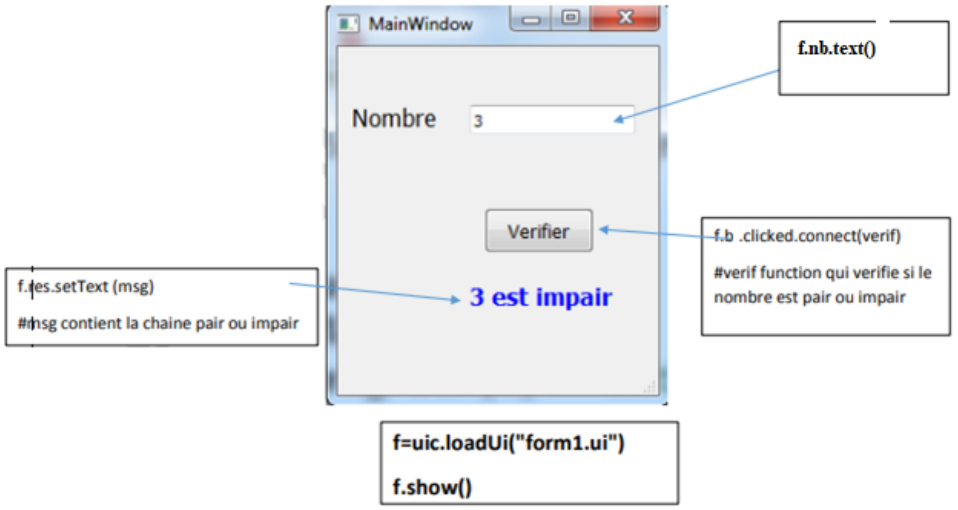
1-Lancer **QtDesigner** et créer l'interface graphique suivante et enregistrer-la sous le nom "**form1**.ui" :

Au niveau QtDesigner



2- Lancer Thonny, puis écrire le programme (et l'enregistrer sous le nom : **parite.py** dans le dossier appelé « **tp3** » dans **C:/bac2023** qui Après avoir remplir la zones nombre et à la suite du clic sur le bouton verifier, le message "n est pair" ou "n est impair" ou sera affiché dans la zone « res » comme le montre l'interface ci-contre

Appel au niveau Python :



```
1  from PyQt5 import QtWidgets,uic
2  ..........
3  ..........
4  ..........
5  ..........
6  ..........
7  ..........
8  ..........
9  ..........
10
11 app=QtWidgets.QApplication([])
12 f=uic.loadUi(" ..... ")
13 f.show()
14 ..........
15 app.exec()
```

Important :

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire Bac2022, créez un dossier de travail ayant comme nom votre numéro d'inscription (6 chiffres) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet

---

Le nombre semi-premier

Un nombre N est dit **semi-premier** lorsqu'il est égal au produit de **deux nombres premiers** non nécessairement distincts. C'est-à-dire N = k*k avec k est un nombre premier ou N = k*j avec k et j sont deux nombres **premiers.**
Exemples :

- ✔ 6 est un nombre semi-premier car 6 = 2 × 3 avec 2 et 3 sont deux nombres premiers.
- ✔ 25 est un nombre semi-premier car 25 = 5 × 5 avec 5 est un nombre premier.
- ✔ 831 est un nombre semi-premier car 831= 3 × 277 avec 3 et 277 sont deux nombres premiers
- ✔ 8 n'est pas un nombre semi-premier, car 8 = 2 × 4 avec 4 n'est pas un nombre premier.

Pour vérifier si un entier naturel N (N > 2) est un nombre semi-premier ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombre semi-premier**"
- Un label demandant la saisie d'un nombre : "**Introduire un entier > 2** : "
- Une zone de saisie permettant la saisie du nombre
- Un bouton intitulé "**Vérifier**"
- Un label pour afficher le message adéquat

Travail demandé :

1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**InterfaceSemiPremier**".
2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**NbrSemiPremier**".

3) Développer, dans le programme "**NbrSemiPremier**", une fonction **SemiPremier**(N) qui permet de vérifier si un entier N est semi-premier ou non.

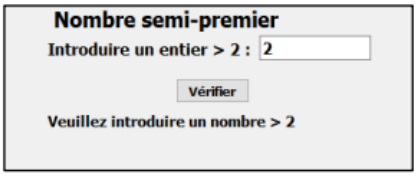
4) Dans le programme "**NbrSemiPremier**" :

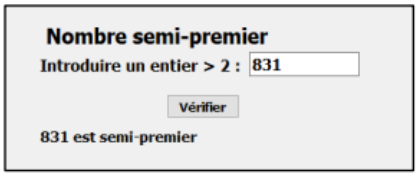
- ✔ Ajouter les instructions permettant d'appeler l'interface graphique intitulée «**InterfaceSemiPremier**" en exploitant l'annexe ci-après.
- ✔ Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Vérifier**", permettant de récupérer l'entier N saisi, puis d'exploiter la fonction "**SemiPremier**" afin d'afficher le message adéquat via le label dédié à l'affichage de l'interface "**InterfaceSemiPremier**".

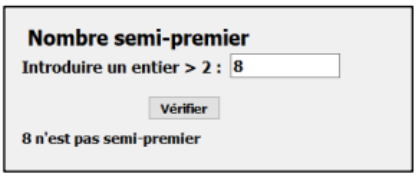
N.B. :

- ✔ Le candidat est appelé à développer un module qui permet de vérifier la primalité d'un entier sans faire recours à des fonctions prédéfinies telles que isprime().
- ✔ L'affichage du message doit être conforme aux exemples d'exécution suivants :

Exemples d'exécution :







Annexe

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
……
……
app = QApplication([])
windows = loadUi ("Nom_Interface.ui")
windows.show()
windows.Nom_Bouton.clicked.connect (Nom_Module)
app.exec_()
```

Grille d'évaluation

Tâches	Nombre de points
Conception de l'interface "**InterfaceSemiPremier**"	**4 pts**
Création et enregistrement du programme "**NbrSemiPremier**"	**1 pt**
Développement de la fonction "**SemiPremier**"	**6 pts**
Ajout des instructions : ▪ de l'interface "**InterfaceSemiPremier**" ▪ du module "**Play**"	**2 pts** **4 pts**
Importation des bibliothèques nécessaires, modularité et cohérence	**3 pts**

Correction :

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
..........................................................
..........................................................
..........................................................
..........................................................
..........................................................

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

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

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


app=QApplication([])
windows=loadUi(" ............................................ ")
windows.show()
..........................................................
app.exec()
```

Les structures de contrôle conditionnelles

La forme simple réduite

Algorithme	Python
Si condition **alors** Traitement **FinSi**	**if** condition : Traitement

La forme alternative

Algorithme	Python
Si condition **alors** Traitement1 **Sinon** Traitement2 **FinSi**	**if** condition : Traitement1 **else :** Traitement2

La forme généralisée

Algorithme	Python
Si condition 1 **alors** traitement 1 **Sinon Si** condition 2 **Alors** traitement 2 **Sinon Si condition 3 Alors** traitement 3 …….. **Sinon Si** condition n-1 **Alors** traitement n-1 **Sinon** traitement n **FinSi**	**if** condition1 : traitement1 **elif** condition2 : traitement2 **elif** condition3 : traitement3 …….. **elif** condition n-1 : traitement n-1 **else :** traitement n

La structure de contrôle conditionnelle à choix

Algorithme	Python
Selon sélecteur **Faire** ***Valeur_1*** : traitement 1 ***Valeur_2*** : traitement 2 ***Valeur_3*** : traitement 3 …….. ***Valeur_n*** : traitement n **Fin selon**	**match** sélecteur : ***case*** valeur 1 : traitement 1 ***case*** valeur2| valeur3 : traitement 2 …….. ***Case _*** : traitement n

Les structures iteratives

La forme complete	
Algorithme	**Python**
Pour i **de** vi à vf **faire** Instruction1 Instruction2 Instruction3 Instruction4 …… Instruction p **Fin pour**	**for i in range(vi , vf) :** Instruction1 Instruction2 Instruction3 Instruction4 …… Instruction p
Répéter jusqu'a	
Algorithme	**Python**
Répéter instruction1 instruction2 …… instruction n **Jusqu'à** condition	**while not condition :** instruction1 instruction2 …… instruction n
Tant que faire	
Algorithme	**Python**
Tant que Condition **faire** Instruction 1 Instruction 2 ….. Instruction N **Fin Tant que**	**while condition :** instruction1 instruction2 …… instruction n

Fonctions et Procédures (Algorithme)

Pour chacun des cas suivants donner l'algorithme d'un sous-programme qui permet de :	
1) Saisir un caractère alphabétique.	**2)** Saisir une chaîne de caractère non vide et de longueur maximale égale à 20.
0) procédure saisie (@ c : caractère)	0) Procedure saisie (@ ch : chaine)
1) Répéter	1) Répéter

```
        Ecrire ("Donner un caractère alphabétique")
        Lire(c)
    Jusqu'à majus(c) dans ["A".."Z"]
2)  Fin saisie
```

3) Vérifier est-ce qu'une chaîne de caractère donnée est alphabétique ou non.

```
0)  Fonction Verif (ch : chaine) : booléen
1)    Test□vrai
2)    i□1
3)    Répéter
        Si NON(majus(ch[i]) dans ["A".."Z"]) Alors
            Test □faux
        FinSi
        i□ i+1
      Jusqu'à (i>long(ch)) ou (test=faux)
4)    Retourner test
5)  Fin Verif
```

5) Remplir un tableau T par N entiers positif d'une manière aléatoire (0 < N ≤100).

```
0)  Procedure Aleatoire(@ t : tab ; @ n : entier)
1)    Repeter
        Ecrire ("Donner 0<n≤100")
        Lire (n)
      Jusqu'à n dans [1..100]
2)    Pour i de 1 à n faire
        T[i]□alea(255)
      Fin Pour
3)  Fin Aleatoire
```

7) Afficher un tableau T de N éléments.

```
0)  Procedure Affiche (T : tab ; n :entier)
1)    Pour i de 1 a n faire
        Ecrire (T[i], " | " )
      Fin Pour
2)  Fin Affiche
```

```
        Ecrire ("Donner une chaine :")
        Lire(ch)
    Jusqu'à long(ch) dans [1..20]
2)  Fin saisie
```

4) Remplir un tableau T par N entiers triés dans l'ordre croissant.

```
0)  Procedure tritab(@ T : tab ; n : entier)
1)    Ecrire ("Donner T[",i, "]= ")
2)    Lire(T[1])
3)    Pour i de 2 a n faire
        Répéter
          Ecrire ("Donner T[",i, "] supérieur à ",T[i-1])
          Lire(T[i])
        Jusqu'à (T[i]>T[i-1])
      FinPour
4)  Fin TriTab
```

6) Compter l'occurrence (nombre d'apparition) d'un caractère dans une cha

```
0)  Fonction Occ(c : caractère ; ch :chaine) :entier
1)    n □ 0
2)    Pour i de 1 à long(ch) faire
        Si ch[i]=c alors
          n□ n+1
        fin si
      Fin Pour
3)    Retourner n
4)  Fin Occ
```

8) Déterminer le maximum d'un tableau.

```
0)  Fonction max (T : tab; n :entier) : entier
1)    M□T[1]
2)    Pour i de 2 à n faire
        Si T[i]> M alors
          M □T[i]
```

```
        FinSi
    Fin Pour
3)  Retourner M
4)  Fin Max
```

9) Inverser une chaîne de caractère.

```
0)  Procedure Inverse (@ ch : chaine)
1)  Pour i de 1 à long(ch) div 2 faire
      X □ Ch[i]
      Ch[i] □ ch[long(ch)-i+1]
      Ch[long(ch)-i+1] □ x
    Fin Pour
2)  Fin Inverse
```

10) Vérifier la présence d'un caractère dans une chaîne.

```
0)  Fonction Present (c:caractère ; ch:chaine):booléen
1)    Test □ faux
2)    i □ 1
3)    Repeter
        Si (ch[i]=c) alors
          Test □ vrai
        FinSi
        i □ i+1
      jusqu'à (i>long(ch)) ou (test=vrai)
4)    Retourner test
5)  Fin Present
```

11) Déterminer le minimum d'un tableau.

```
0)  Fonction min (T : tab; n :entier) : entier
1)    M□T[1]
2)    Pour i de 2 à n faire
        Si T[i]< M alors
          M □T[i]
        FinSi
      Fin Pour
3)    Retourner Min
4)  Fin Min
```

12) Vérifier est-ce qu'une chaîne de caractère donnée est numérique ou non.

```
0)  Fonction Verif (ch : chaine) : booléen
1)    i□1
2)    Répéter
        Test □ majus(ch[i]) dans ["A".."Z"]
        i□ i+1
      Jusqu'à (i>long(ch)) ou (test=faux)
3)    Retourner  test
4)  Fin Verif
```


Pour chacun des cas suivants donner l'algorithme d'un sous-programme qui permet de :

1) Saisir un caractère alphabétique.

```
def saisie () :
```

2) Saisir une chaîne de caractère non vide et de longueur maximale égale à 20.

```
def saisie () :
```

```
    c = input ("Donner un caractère alphabétique: ")
    while(not('A'<=c.upper()<='Z')):
        c = input ("Donner un caractère alphabétique: ")
    return c
#-----Prog. Principal----
c=saisie()
```

```
    ch = input ("Donner une chaine de caractère: ")
    while(not(len(ch) in range(1,20))):
        ch = input ("Donner une chaine de caractère : ")
    return ch
#-----Prog. Principal----
ch=saisie()
```

3) Vérifier est-ce qu'une chaîne de caractère donnée est alphabétique ou non.

```
def verif (ch ) :
    test=True
    i=1
    while(i<len(ch)) and (test==True):
        if not('A'<=ch[i].upper()<='Z') :
            test=False
        i=i+1
    return test



#-----Prog. Principal----
ch = input ("Donner une chaine de caractere: ")
b=verif(ch)
print(b)
```

4) Remplir un tableau T par N entiers triés dans l'ordre croissant.

```
from numpy import *
T=array([int]*10)
def Remplir_ordre( n):
    print ("Donner T[0] :",end="")
    T[0]=int(input())
    for i in range(1,n) :
     T[i]=int(input('Donner T['+str(i)+'] :'))
     while T[i]<T[i-1] :
       print ("Donner T[" , i , "] : ",end="")
       T[i]=int(input('Donner T['+str(i)+'] :'))
    return T

#-----Prog. Principal----
n=int(input('donner n'))
T=Remplir_ordre(n)
for i in range(n):
    print(T[i])
```

5) Remplir un tableau T par N entiers positif d'une manière aléatoire (0 < N ≤100).

```
from numpy import *
t=array([int]*100)
def  remplir():
 print ("Donner n : ",end="")
```

6) Compter l'occurrence d'un caractère dans une chaîne.

```
def occurence(c,ch):
 nb=0
 for i in range(len(ch)):
      if ch[i] ==c :
```

```
 n=int(input())
 while n <=0 or n>100:
    print ("Donner n : ",end="")
    n=int(input())
 for i in range (n) :
  print ("Donner t[" , i , "] : ",end="")
  t[i]=int(input())
  while t[i] <0 :
    print ("Donner t[" , i , "] : ",end="")
    t[i]=int(input())
 return n,t
  #-----Prog. Principal----
n,t=remplir()
for i in range(n):
    print(t[i])
```

7) Afficher un tableau T de N éléments.

```
from numpy import *
t=array([int]*10)
def remplir( n):
 for i in range(0,n) :
    t[i]=int(input("Donner T[" + str(i) + "] : "))

 return t
def afficher( n,t):
 for i in range(0,n) :
   print( t[i])

n=int(input('donner la taille'))
t=remplir(n)
afficher(n,t)
```

9) Déterminer le minimum d'un tableau.

```
from numpy import *
t=array([int]*10)
```

```
            nb=nb+1
 return nb
c=input('donner un caractere')
ch=input('donner une chaine')
n=occurence(c,ch)
print(n)
```

8) Déterminer le maximum d'un tableau.

```
from numpy import *
t=array([int]*10)
def remplir( N):
 for i in range(0,N) :
    t[i]=int(input("Donner T[" + str(i) + "] : "))
 return t
def maximum(t,N):
     max=t[0]
     for i in range(1,N):
        # Si l'élément actuel est supérieur à max */
            if (t[i] > max):
                    max = t[i]
     return max
N=int(input('donner n'))
t=remplir(N)
print(maximum(t,N))
```

10) Vérifier la présence d'un caractère dans une chaîne.

```
def Present (c, ch) :
 trouve=False
```

```
def remplir( n):
 for i in range(0,n) :
   t[i]=int(input("Donner T[" + str(i) + "] : "))

 return t
def minimum(t,N):
   # Supposons le premier élément comme minimum */
    min=t[0]
    for i in range(1,N):
      # Si l'élément actuel est inferieur à min */
         if (t[i]< min):
                min = t[i]
    return min
#-----Prog. Principal----
N=int(input('donner n'))
t=remplir(N)
     print(minimum(t,N))
```

```
i=0
while(i<len(ch)) and (trouve==False):
 if ch[i]==c :
    trouve=True
 else :
     i=i+1

if trouve == True :
 msg="existe"
else :
 msg="n'existe pas"
return msg
c=input('donner c')
ch=input('donner ch')
msg=Present(c,ch)
print(msg)
```

11)Vérifier est-ce qu'une chaîne de caractère donnée est numérique ou non.

```
def verif (ch) :
 test=True
 i=0
 while(i<len(ch)) and (test==True):
  if not('A'<=ch[i].upper()<='Z'):
     test=False
  else :
      i=i+1
 return test
ch=input('donner ch')
v=verif(ch)
print(v)
```

Procédure saisir n avec contrôle de saisie :

```
Procédure saisir (@m:entier )
Début
  Répéter
      Ecrire ("m=");Lire( m )
  Jusqu'à (m dans[5..10]);
Fin
```

Procédure remplissage d'un tableau avec condition :
(remplissage par des entiers >0)

```
Procédure remplir (@t:tab;n:entier )
Début
Pour i de 1 à n-1 faire
          Répéter
            Ecrire ("t[",i,"] = ");
            Lire (t[i])
          Jusqu'à (t[i]>0)
Fin Pour
Fin
```

	ontrôle de saisie: Procédure empêcher (@v:	

	tab;m:entier)Début Pour i de 0 à n-1 faire	

```
Ecrire("v[",i,"]=");
Lire(v[i]);
Finpo
```

	urfin	
Procédure remplissage d'un tableau par des entiers au hasard (entre 10 et 90) : Procédure remplir (@t:tab;n:entier) Début Pour i de 1 à n-1 faire t[i] aléa (10,90) fin pour Fin	**Procédure remplissage d'un tableau par des éléments en ordre :**	Procédure remplir (@t:tab;n:entier) Début Ecrire ("T[",0,"] = ") ; Lire (t[0]) Pour i de 1 à n-1 faire **Répéter** Ecrire ("t[",i,"] = "); Lire (t[i]) **Jusqu'à (t[i]>t[i-1])** Fin Pour Fin

	par desletttresalphabétiquemajusculeauh	

	asard Procédure remplir(@t:tab;n:entier	

r)Début Pour i de 1 à n-1 faire t[i] □ chr(al

	é a (o r d (' A ') , o r d (' Z ')) F i n p o u r f i n	
Procédure remplissage d'un tableau **par des entiers de deux chiffres** Procédure remplir (@t:tab;n:entier) Début Pour i de 1 à n-1 faire	**P** **r** **o** **c** **é**	**Procédure remplissage d'un tableau** **par des chaines ne contenant que des lettres alphabétiques** Procédure remplir (@t:tab;n:entier) Début Pour i de 1 à n-1 faire

```
        Répéter
            Ecrire ("t[",i,"] = ");
            Lire (t[i])
        Jusqu'à (10<=t[i]<=99)
Fin Pour
Fin
```

duremplissaged'untableaupardessentiere

```
        Répéter
            Ecrire ("t[",i,"] = ");
            Lire (t[i])
        Jusqu'à (verif(t[i])=vrai)
Fin Pour
Fin
```

	r s d i s t i n c t P r o c é d u r e r e m p l i r (@ t : t a b ; n : e	

	ntier)DébutPouridel1àn-1faire	

	FinPourFin	

Fonction nombre occurrence de x dans tableau T de taille n :

```
Fonction nbocc(t :tab,n,x :entier):entier
Début
  nb 0
  Pour i de 0 à n-1 faire
    Si t[i]=x alors
       nbnb+1
    fin si
  fin pour
  retourner (nb)
fin
```

Fonction recherche d'un élément x dans un tableau

Fonction verif pour verifier si la chaine ch ne contient que des lettres alphabétiques

```
fonction verif(ch :chaine):booléen
Début
   i0
   test vrai
   tant que (i<long(ch))et (test==vrai) faire
      si ('A'< majus(ch[i])< 'Z') alors
          ii+1
      sinon
          testFaux
       finsi
   fin tantque
   retourner( test)
fin
```

	bleauTdenélément **s** fonctionrechercher(T	

	:tab,n,x:entier):booléenDébut i□0 tro	

	uve Faux tantque(i<n) et(trouve==Faux)	

	faire siv[i]=yalors trouve□Vrai sinon	

	i □ i + 1 f i n s i f i n t a n t q u e r e t o u r n e r (t r o	

	uve)fin	
Fonction maximum d'un tableau : fonction Maximum(t:tab;n:entier) : entier Début Max □ t[0] Pour i de 1 à n-1 faire Si t[i] > Max alors Max □ t[i] Finsi Fin pour Retourner(Max) Fin	**Fonction inverse d'une chaine ch**	**Fonction somme des éléments de tableau T de n entiers :** fonction somme (t:tab;n:entier) : entier Debut S □0 Pour i de 0 à n-1 faire S □S + t[i] Fin pour Retourner S Fin

	f o n c t i o n i n v e r s e (c h : c h a i n e) : c h a i n e D é	

	but ch1□,,,, Pour ide0allong(ch)-1faire	

ch1 ch[i]+ch1

Finpour retourner(ch1)

fin

Fonction somme des diviseurs d'un entier x :

```
fonction sommediv (x :entier ) : entier
Début
   S □0
   Pour i de 1 à x faire
   Si x mod i=0 alors
        S □S + i
   Fin pour
   Retourner S
Fin
```

Fonction somme des chiffres d'un entier N

```
Fonction sommechiffre(N:entier):entire
début
   S □ 0
   Répéter
        S □S + N MOD 10
        N □N DIV 10
   Jusqu'à N = 0
   Retourner (S)
fin
```

Fonction nombre des diviseurs d'un entier

	ierx: Fonctionnbdiv(x :entier):entier	

	Début nb 0 Pour i de 1 à x faire Si x mod i = 0 a	

	lors nb nb+i Finpour Retourner nb Fin	

Procédure affichage d'un tableau :

```
Procédure affichage( v:tab;n:entier)
Début
  Pour i de 0 à n-1 faire
    Ecrire ("V[",i,"] = ",v[i]);
  Finpour
fin
```

Procédure extraire à partir d' un tableau T les éléments pairs dans Tp et impairs dans Timp

```
Procédure extraire(t,@TP,@Timp:tab;n,@j,@k:entier )
Début
   j □ 0
   k□0
   Pour i de 0 à n-1 faire
        Si(t[i] mod 2 = 0) alors
            Tp[j]□ t[i] ,
            j□j+1
        Sinon
            Timp □t[i]
            k□ k+1
         Finsi
    Fin pour
Fin
```

NB :le tableau TP est de taille j
Le tableau Timp est de taille k

Procédure d'un tableau avec cond

	i t i o n : (a f f i c h e r l e s é l é m e n t s p a i r s d e T) Pr	

	o c é d u r e a f f i c h e (t : t a b ; n : e n t i e r) D é b u t P	

	ouri de0 àn- 1f air e sit[i]m od2 =0 al ors Ec	

```
rire("T["",i,""]="",t[i]);' finsi Finpour
```

Fin

Procédure insertion d'un élément x dans un tableau T à une position p

```
Procédure insertion(@v : tab ;n,x,p : entier):
Début
     Pour i de  n à p+1 (pas= -1) faire
           t[i]=t[i-1]
    fin pour
    t[p]=x
fin
```

NB : pour insérer un nouvel élément dans T on doit :

1) décaler les éléments de T vers la droite pour vider la position p

2) insérer x dans sa position p

retourner v

Fin

Fonction ppcm(a,b :entier):entier

```
Début
   x□a
   Tantque (x mod b≠0) faire
       x□x+a
   fin tantque
   retourner(x)
fin
```

	d é b u t T a n t q u e (a ≠ b) f a i r e s i (a > b) a l o r s	

	a □ a - b s i n o n b □ b - a f i n s i f i n t a n t q u e r	

	e t o u r n e r (a) **f i n**	