

**Qt**

Kais TABOUBI

2022-2023

SUCCESS

25 Sujets

Pour bien préparer  
au bac pratique

## Sections:

Mathématiques, Sciences expérimentales  
et Sciences techniques

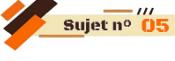
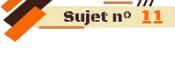
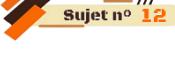
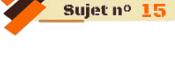
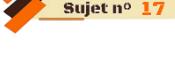
# REVISION BAC PRATIQUE 2023

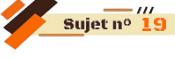
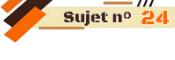
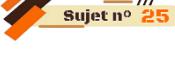
MATIERE : INFORMATIQUE

ENSEIGNANT : Kais TABOUBI

\*\*\*\*\*

## SOMMAIRE

 Sujet n° 01	Graphique **** Chariot gratuit payant	4
 Sujet n° 02	Graphique **** Cryptographie par alphabet-Clé	7
 Sujet n° 03	Graphique **** Cryptographie par occurrence successive	10
 Sujet n° 04	Graphique **** Cryptographie par apparition	13
 Sujet n° 05	Graphique **** Mots Frères-Cousins	16
 Sujet n° 06	Graphique **** Mot presque palindrome	19
 Sujet n° 07	Graphique **** Arithmétique_Nombre Pointu	22
 Sujet n° 08	Graphique **** Arithmétique_Nombre Spécial	25
 Sujet n° 09	Graphique **** Arithmétique_Nombre totalement impair	28
 Sujet n° 10	Graphique **** Arithmétique_Nombre doublement palindrome	31
 Sujet n° 11	Graphique **** Arithmétique_Nombre de Smith	34
 Sujet n° 12	Numpy **** **** Nombres-Sectionnable unitaire	37
 Sujet n° 13	Numpy **** **** Nombres Etranges	38
 Sujet n° 14	Numpy **** **** Nombres doublement palindrome	39
 Sujet n° 15	Numpy **** **** Concours d'une commune	41
 Sujet n° 16	Graphique **** Arithmétique_CRODEC DECRO	43
 Sujet n° 17	Graphique **** Emprisonnement d'un mot	45
 Sujet n° 18	Graphique **** Arithmétique_Nombres Homogènes	48

 <b>Sujet n° 19</b>	<b>Graphique **** Arithmétique_PPMB</b>	<b>51</b>
 <b>Sujet n° 20</b>	<b>Graphique **** Arithmétique_Nombre Multipotent</b>	<b>54</b>
 <b>Sujet n° 21</b>	<b>Graphique **** Arithmétique_Nombre Lisse</b>	<b>57</b>
 <b>Sujet n° 22</b>	<b>Graphique **** Arithmétique_Nombre semi premier</b>	<b>59</b>
 <b>Sujet n° 23</b>	<b>Graphique **** Arithmétique_Nombre hautement abondant</b>	<b>61</b>
 <b>Sujet n° 24</b>	<b>Graphique **** Prototype 2022</b>	<b>63</b>
 <b>Sujet n° 25</b>	<b>Graphique **** Salle de Sport</b>	<b>66</b>

**Important** 1. Une solution modulaire au problème posé est exigée.

2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

Dans le 100<sup>ème</sup> anniversaire, une entreprise commerciale décide, à la caisse, d'offrir aux clients leurs achats. Pour cela, et suite aux règlements de l'offre, le caissier demande du client son numéro de téléphone **N** (contient **8** chiffres), dont son premier doit être un parmi la liste suivante (**2, 3, 4, 5 et 9**).

On se propose d'afficher si le client qui a le numéro de téléphone **N** est gagnant ou non. Un client est déclaré gagnant si le **carré** du chiffre de chance **CC** de son numéro de téléphone existe dans un tableau **T** déjà rempli aléatoirement par **8** entiers appartenant à l'intervalle **[1, 99]**.

Un chiffre de chance **CC** relatif à un numéro de téléphone est calculé en additionnant de façon répétitive tous les chiffres qui composent le numéro de téléphone jusqu'à obtenir un seul chiffre.

**Exemples :**

❶ Pour le numéro de téléphone **N** et le tableau **T** suivants: **N = 55 405 103**

T	13	7	19	25	93	64	7	1
	0	1	2	3	4	5	6	7

Le programme affiche " *Le client ayant le numéro de téléphone 55 405 103 a gagné un chariot gratuit* ".

**En effet**, le **CC** du client a été obtenu en additionnant les chiffres de son numéro de téléphone jusqu'à obtenir un seul chiffre c'est à dire : \*  $5+5+4+0+5+1+0+3 = 23$

$$** \quad 2+3 = 5.$$

Le carré du chiffre **5**, ( $5^2=25$ ), figure dans le tableau **T**, donc c'est un client gagnant.

❷ Pour le numéro de téléphone **N** et le tableau **T** suivants: **N = 99 678 987**

T	8	54	96	38	54	5	11	54
	0	1	2	3	4	5	6	7

Le programme affiche " *Le client ayant le numéro de téléphone 99 678 987 doit payer les achats du chariot* ".

**En effet**, le **CC** du client a été obtenu en additionnant les chiffres de son numéro de téléphone jusqu'à obtenir un seul chiffre c'est à dire : \*  $9+9+6+7+8+9+8+5 = 55$

$$** \quad 5+5 = 10$$

$$*** \quad 1+0 = 1.$$

Le carré du chiffre **1**, ( $1^2=1$ ), ne figure pas dans le tableau **T**, donc c'est un client perdant.

Ci-après, la procédure **REMP\_AFFICHE** à exploiter pour résoudre le problème posé, qui permet de remplir aléatoirement le tableau **T**, de type TAB, par **8** entiers dans l'intervalle **[1, 99]** afin de les afficher sur la console.

```

Procédure REMP_AFFICHE (@T : TAB)
  Début
    Pour i de 0 à 7 faire
      T[i]  $\leftarrow$  Aléa(1, 99)
      Ecrire (T[i], " | ")
    Fin Pour
  Fin
```

**NB** : l'affichage des éléments du tableau **T** dans la console vous aide pour tester votre programme lors de la saisie du numéro de téléphone du client (gain ou perd).

On se propose de concevoir une interface graphique contenant les éléments suivants :

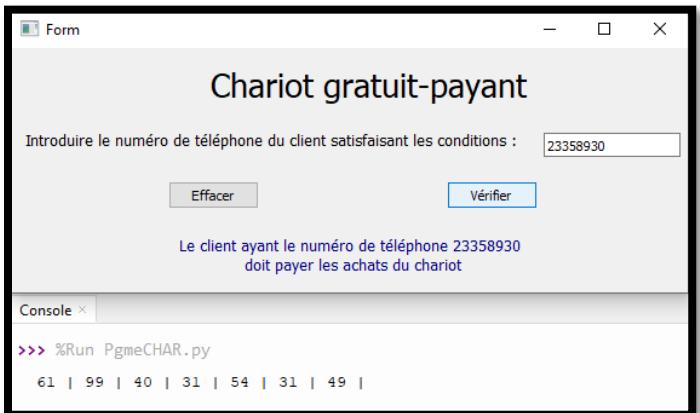
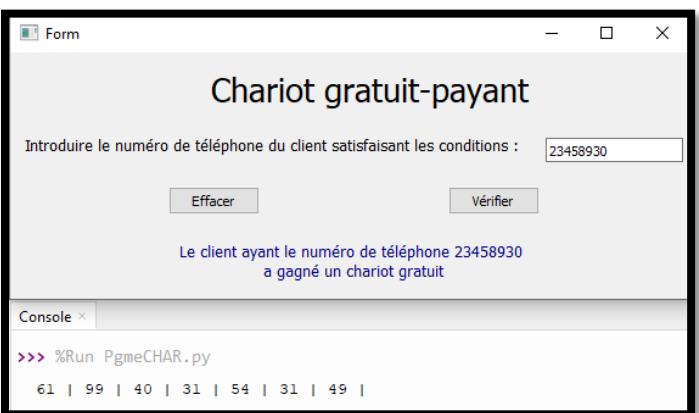
- Un label contenant le texte : "**Chariot gratuit-payant**"
- Un label demandant la saisie du nombre :  
"Introduire le numéro de téléphone du client satisfaisant les conditions :"
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé " **Valider**"
- Un bouton intitulé " **Effacer**" (*permettant d'effacer la zone de saisie et le label d'affichage*)
- 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 "**InterCHARIOT**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeCHAR**".
- 3) Implémenter la procédure **REMP\_AFFICHE** et l'appeler dans le programme principal.
- 4) Développer dans le programme "**PgmeCHAR**", une fonction **GainPerd (N)** qui permet de vérifier si le numéro de téléphone du client **N** est gagnant ou non.
- 5) Dans le programme "**PgmeCHAR**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterCHARIOT**" en exploitant l'**annexe** ci-après.
  - Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Valider**", permettant de récupérer l'entier **N** saisi satisfaisant les conditions décrites précédemment, puis d'exploiter la fonction **GainPerd** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterCHARIOT**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :



## 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 <b>InterCHARIOT</b>	<b>3 pts</b>
Création et enregistrement du programme <b>PgmeCHAR</b>	<b>1 pt</b>
Développement de la fonction <b>GainPerd</b>	<b>6 pts</b>
Ajout des instructions : * De l'interface <b>InterCHARIOT</b> * Du module <b>Play</b>	<b>1 pt</b> <b>3 pts</b>
Importation des bibliothèques nécessaires, modularité et cohérence	<b>3 pts</b>
Implémentation de la procédure <b>REMP_AFFICHE</b> et son appel	<b>3 pts</b>

**Important** 1. Une solution modulaire au problème posé est exigée.

2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

Une technique ultérieure de cryptographie consista à opérer non avec un décalage systématique, mais par une substitution aléatoire. Pour cela, on utilise un alphabet-clé, dans lequel les lettres se succèdent de manière désordonnée, par exemple : **HYLUJPVREAKBNDOFSQZCWMGITX**

C'est cette clé qui va servir ensuite à coder le message. Selon notre exemple, les **A** deviendront des **H**, les **B** des **Y**, les **C** des **L**, etc.

On se propose de saisir un message non nul contenant uniquement des lettres **alphabétiques majuscules** et des **espaces** puis d'effectuer et d'afficher son cryptage (l'alphabet-clé sera générer aléatoirement).

### Exemple :

Soit le tableau ci-dessous représente les **ordres des lettres alphabétiques majuscules** :

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

① Pour le message **MES** et l'**alphabet-clé** suivants:

**MES = "LE CRYPTAGE DES INFORMATIONS"**

L'**alphabet-clé** : 

W	R	L	J	G	T	D	P	K	Z	M	X	C	F	S	H	Y	B	O	A	E	U	V	Q	I	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Le programme affiche "**XG LBIHAWDG JGO KFTSBCWAKSFO**".

- La lettre "L" sera cryptée par "X". **En effet**, l'ordre de la lettre "L" est égal à **11** dans l'ensemble des lettres alphabétiques, alors cette lettre sera cryptée par son équivalent dans l'**alphabet-clé** qui est la lettre "X".
- La lettre "E" sera cryptée par "G". **En effet**, l'ordre de la lettre "E" est égal à **4** dans l'ensemble des lettres alphabétiques, alors cette lettre sera cryptée par son équivalent dans l'**alphabet-clé** qui est la lettre "G".
- **Et ainsi de suite**....

Ci-après, la déclaration **incomplète** de la fonction **Generer** à exploiter pour résoudre le problème posé, qui permet de générer aléatoirement une chaîne contenant les **26** lettres alphabétiques majuscules.

```

1 from random import randint
2 def generer():
3     ch=""
4     for i in range(26):
5         n=randint(65,90)
6         while ..... :
7             n=randint(65,90)
8             ch+= .....
9     return ch

```

#### Question :

Pour compléter les pointillés il suffit de répondre aux tâches suivantes :

- ① **La ligne n°6** : On répète la génération aléatoire du nombre tant que le caractère équivalent à **n** existe dans la chaîne **ch**.
- ② **La ligne n°8** : La concaténation de la chaîne **ch** avec le caractère équivalent à **n**.

On se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "Cryptographie avec l'alphabet-Clé"
- Un label demandant la saisie du message :  
"Introduire un message dont sa longueur appartient à l'intervalle [1, 100] \n et contient que des lettres majuscules et des espaces :"
- Une zone de saisie permettant la saisie du message.
- Un bouton intitulé "Crypter"
- Un bouton intitulé "Effacer" (*permettant d'effacer la zone de saisie et les deux labels d'affichage*)
- Un label permettant d'afficher la clé (résultat retourner par la fonction **Generer()**)
- 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 "**InterCRYPTE\_AC**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeCRYPTE\_AC**".
- 3) Implémenter la fonction **Generer** tout en complétant les pointillés en répondant sur la question posée précédemment.
- 4) Développer dans le programme "**PgmeCRYPTE\_AC**", une fonction **Crypter (m, cle)** qui permet de crypter le message **m** en utilisant **cle** (résultat retourner par la fonction **Generer()**).
- 5) Dans le programme "**PgmeCRYPTE\_AC**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterCRYPTE\_AC**" en exploitant l'**annexe** ci-après.
  - Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Crypter**", permettant de récupérer le message **m** saisi satisfaisant les conditions décrites précédemment, puis d'exploiter la fonction **Crypter** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterCRYPTE\_AC**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

**Exemples d'exécution**

**Cryptographie par l'alphabet-Clé**

Introduire un message dont sa longueur appartient à l'intervalle [1,100] et contient que des lettres majuscules et des espaces :

Effacer

Crypter

Désolé ! vérifier le message à crypter !!

**Cryptographie par l'alphabet-Clé**

Introduire un message dont sa longueur appartient à l'intervalle [1,100] et contient que des lettres majuscules et des espaces :

Effacer

Crypter

Désolé ! vérifier le message à crypter !!

**Cryptographie par l'alphabet-Clé**

Introduire un message dont sa longueur appartient à l'intervalle [1,100] et contient que des lettres majuscules et des espaces :

Effacer

Crypter

la clé générée est: OPDKNQBGYRZVSWLFHJCATUIXM

Le message crypté sera: OVY UO OT DYWNSO

**Cryptographie par l'alphabet-Clé**

Introduire un message dont sa longueur appartient à l'intervalle [1,100] et contient que des lettres majuscules et des espaces :

Effacer

Crypter

la clé générée est: XLYQPGUROHBZDISEKV/MACWFNJT

Le message crypté sera: XZO WX XC YOIPDX

## 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 <b>InterCRYPTE_AC</b>	3 pts
Création et enregistrement du programme <b>PgmeCRYPTE_AC</b>	1 pt
Développement de la fonction <b>Crypter</b>	6 pts
Ajout des instructions : * De l'interface <b>InterCRYPTE_AC</b> * Du module <b>Play</b>	1 pt 3 pts
Importation des bibliothèques nécessaires, modularité et cohérence	3 pts
Implémentation de la fonction <b>Generer</b> en complétant les pointillés	3 pts

**Important** 1. Une solution modulaire au problème posé est exigée.

2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

On veut crypter une chaîne de caractères données **CH** dont la taille ne dépasse pas **50** caractères en une chaîne résultat **CHC** de la manière suivante :

- Parcourir la chaîne **CH** de gauche à droite en comptant le nombre **d'occurrences successives** de chaque caractère de la chaîne **CH**,
- Puis de ranger la chaîne **CHC**, ce nombre suivi du caractère en question.

On se propose de saisir une chaîne de caractères **CH** qui doit être **non vide** et formée uniquement par **des lettres alphabétiques**, puis de former et d'afficher la chaîne **CHC** selon le principe décrit précédemment.

### Exemple :

① Pour la chaîne de caractères **CH** = "aaaFyBssssssssssssazz"

Le programme affiche **CHC** : "3a1F1y1B12s1a2z".

- En effet, on a 3 occurrences successives de la lettre "a", d'où la première partie de **CHC** contient "3a"
- Ensuite, on a une seule occurrence de la lettre "F", d'où la chaîne **CHC** devient "3a1F"
- Et ainsi de suite,...

On se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Cryptographie avec occurrence successive**"
- Un label demandant la saisie de la chaîne :
 

"Introduire une chaîne dont sa longueur appartient à l'intervalle [1, 50] \n et contient que des lettres alphabétiques :"
- Une zone de saisie permettant la saisie de la chaîne.
- Un bouton intitulé "Crypter"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- Un label pour afficher le message adéquat.

**Cryptographie par occurrence successive**

Introduire une chaîne dont sa longueur appartient à l'intervalle [1, 50]  
et contient que des lettres alphabétiques :

Effacer
Crypter

## Travail demandé :

- 1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**InterCRYPTE\_OS**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeCRYPTE\_OS**".
- 3) Développer dans le programme "**PgmeCRYPTE\_OS**", une fonction **Crypter (CH)** qui permet de crypter la chaîne **CH** en utilisant le principe décrit précédemment.
- 4) Dans le programme "**PgmeCRYPTE\_OS**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterCRYPTE\_OS**" en exploitant l'**annexe** ci-après.
  - Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Crypter**", permettant de récupérer la chaîne **CH** saisi satisfaisant les conditions décrites précédemment, puis d'exploiter la fonction **Crypter** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterCRYPTE\_OS**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

Exemples d'exécution

### Cryptographie par occurrence successive

Introduire une chaîne dont sa longueur appartient à l'intervalle [1, 50] et contient que des lettres alphabétiques :

Effacer

Crypter

Désolé ! vérifier la chaîne à crypter !!

### Cryptographie par occurrence successive

Introduire une chaîne dont sa longueur appartient à l'intervalle [1, 50] et contient que des lettres alphabétiques :

Effacer

Crypter

Désolé ! vérifier la chaîne à crypter !!

### Cryptographie par occurrence successive

Introduire une chaîne dont sa longueur appartient à l'intervalle [1, 50] et contient que des lettres alphabétiques :

Effacer

Crypter

La chaîne cryptée sera: 3a1F1y1B12s1a2z

### Cryptographie par occurrence successive

Introduire une chaîne dont sa longueur appartient à l'intervalle [1, 50] et contient que des lettres alphabétiques :

Effacer

Crypter

La chaîne cryptée sera: 2B4A2C

## 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 <b>InterCRYPTE_OS</b>	<b>3 pts</b>
Création et enregistrement du programme <b>PgmeCRYPTE_OS</b>	<b>1 pt</b>
Développement de la fonction <b>Crypter</b>	<b>6 pts</b>
Ajout des instructions : * De l'interface <b>InterCRYPTE_OS</b> * Du module <b>Play</b>	<b>2 pt</b> <b>5 pts</b>
Importation des bibliothèques nécessaires, modularité et cohérence	<b>3 pts</b>

**Important** 1. Une solution modulaire au problème posé est exigée.

2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

On veut crypter un mot saisi **MS** non vide, composé uniquement par des lettres majuscules et d'afficher le mot crypté **MC**.

La méthode de cryptage est la suivante :

- Pour chaque lettre, déterminer son nombre d'occurrence (apparition) **n** dans le mot **MS**.
- Déterminer **K** qui est égal à  $2*n$  si **n** est impair et sera égal à  $(n \text{ DIV } 2)$  si **n** est pair.
- Remplacer chaque lettre par **K**<sup>ème</sup> lettre qui la suit dans l'intervalle de l'alphabet ['A'..'Z']. Pour les dernières lettres, on rend dès le début, par exemple si **K=3**, on remplacera 'A' par 'D', 'B' par 'E', 'C' par 'F'... 'Y' par 'B' et 'Z' par 'C'.

### Exemple :

① Pour le mot **MS = "ZEBRE"**

	"Z"	"E"	"B"	"R"	"E"
Nombre d'apparition (n)	1	2	1	1	2
La valeur de K	$2*1 = 2$	$2 \text{ DIV } 2 = 1$	$2*1 = 2$	$2*1 = 2$	$2 \text{ DIV } 2 = 1$
La lettre de remplacement	"B"	"F"	"D"	"T"	"F"

Le programme affiche **MC : "BFDTF"**.

On se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Cryptographie par apparition dans le mot**"
- Un label demandant la saisie du mot:  
"**Introduire un mot non vide et composé uniquement par des lettres majuscules :**"
- Une zone de saisie permettant la saisie du mot.
- Un bouton intitulé "**Crypter**"
- Un bouton intitulé "**Effacer**" (permettant d'effacer la zone de saisie et le label d'affichage)
- Un label pour afficher le message adéquat.

**Cryptographie par apparition dans le mot**

Introduire un mot non vide et composé uniquement par des lettres majuscules :

Effacer
Crypter

## Travail demandé :

- 1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**InterCRYPTE\_AM**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeCRYPTE\_AM**".
- 3) Développer dans le programme "**PgmeCRYPTE\_AM**", une fonction **Crypter (MS)** qui permet de crypter le mot **MS** en utilisant **le principe décrit précédemment**.
- 4) Dans le programme "**PgmeCRYPTE\_AM**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterCRYPTE\_AM**" en exploitant **l'annexe** ci-après.
  - Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Crypter**", permettant de récupérer le mot **MS** saisi satisfaisant les conditions décrites précédemment, puis d'exploiter la fonction **Crypter** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterCRYPTE\_AM**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

**Exemples d'exécution**

**Cryptographie par apparition dans le mot**

Introduire un mot non vide et composé uniquement par des lettres majuscules :

Effacer Crypter

Désolé ! vérifier le mot à crypter !!

**Cryptographie par apparition dans le mot**

Introduire un mot non vide et composé uniquement par des lettres majuscules :

Effacer Crypter

Le mot crypté sera: BFDTF

**Cryptographie par apparition dans le mot**

Introduire un mot non vide et composé uniquement par des lettres majuscules :

Effacer Crypter

Le mot crypté sera: RSPISBNNBVKPP

## 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 <b>InterCRYPTE_AM</b>	<b>3 pts</b>
Création et enregistrement du programme <b>PgmeCRYPTE_AM</b>	<b>1 pt</b>
Développement de la fonction <b>Crypter</b>	<b>6 pts</b>
Ajout des instructions : * De l'interface <b>InterCRYPTE_AM</b> * Du module <b>Play</b>	<b>2 pt</b> <b>5 pts</b>
Importation des bibliothèques nécessaires, modularité et cohérence	<b>3 pts</b>

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

**Les mots frères-cousins**

□ On autorise deux règles de transformation des mots :

- **Règle1** : si toute consonne dans **MotA** est remplacée par une consonne dans **MotB** et toute voyelle dans **MotA** est remplacée par une voyelle dans **MotB**. On dira que **MotA** est frère de **MotB**.
- **Règle2** : si toute consonne dans **MotA** est remplacée par une voyelle dans **MotB** et toute voyelle dans **MotA** est remplacée par une consonne dans **MotB**. On dira que **MotA** est cousin de **MotB**.

**NB :** \* Les deux mots ne doivent pas être *vides* et contiennent que des *lettres alphabétiques minuscules*.

\*\* La longueur du **MotA** est égale à celle du **MotB**.

**Exemples :**

① **MotA = "chlore"** est frère de **MotB="frwihu"** car:

- La 1<sup>ère</sup> lettre du **MotA** : "c" (**consonne**) sera remplacée par "f" (**consonne**) dans **MotB**,
- La 2<sup>ème</sup> lettre du **MotA** : "h" (**consonne**) sera remplacée par "r" (**consonne**) dans **MotB**,
- La 3<sup>ème</sup> lettre du **MotA** : "l" (**consonne**) sera remplacée par "w" (**consonne**) dans **MotB**,
- La 4<sup>ème</sup> lettre du **MotA** : "o" (**voyelle**) sera remplacée par "i" (**voyelle**) dans **MotB**,
- La 5<sup>ème</sup> lettre du **MotA** : "r" (**consonne**) sera remplacée par "h" (**consonne**) dans **MotB**,
- La 6<sup>ème</sup> lettre du **MotA** : "e" (**voyelle**) sera remplacée par "u" (**voyelle**) dans **MotB**,

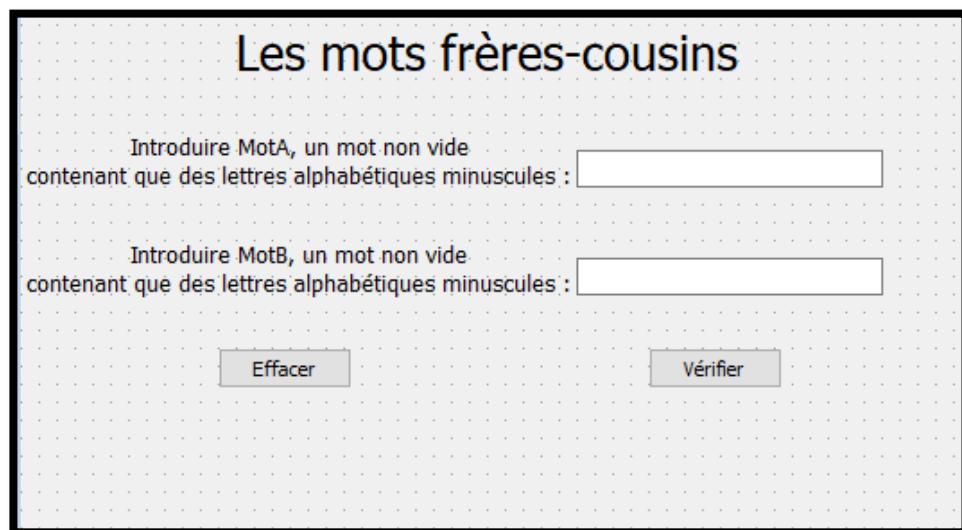
② **MotA = "bon"** est cousin de **MotB="ali"** car:

- La 1<sup>ère</sup> lettre du **MotA** : "b" (**consonne**) sera remplacée par "a" (**voyelle**) dans **MotB**,
- La 2<sup>ème</sup> lettre du **MotA** : "o" (**voyelle**) sera remplacée par "l" (**consonne**) dans **MotB**,
- La 3<sup>ème</sup> lettre du **MotA** : "n" (**consonne**) sera remplacée par "i" (**voyelle**) dans **MotB**,

③ **MotA = "moez"** et **MotB="sara"** sont ni frères ni cousins car ils ne vérifient ni la **règle1** ni la **règle2**

Pour déterminer le type de deux mots **MotA** et **MotB** (**frères**, **cousins** ou bien **ni frères ni cousins**), on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Les mots frères-cousins**"
- Un label demandant la saisie du premier mot :  
"Introduire MotA, un mot non vide contenant que des lettres alphabétiques minuscules : "
- Une zone de saisie permettant la saisie du **MotA**.
- Un label demandant la saisie du second mot :  
"Introduire MotB, un mot non vide contenant que des lettres alphabétiques minuscules : "
- Une zone de saisie permettant la saisie du **MotB**.
- Un bouton intitulé " **Vérifier**"
- Un bouton intitulé " **Effacer**" (*permettant d'effacer la zone de saisie et le label d'affichage*)
- 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 "**InterFRCO**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeFRCO**".
- 3) Développer dans le programme "**PgmeFRCO**", deux fonctions **Freres(MotA, MotB)** et **Cousins(MotA, MotB)** qui permettent de vérifier respectivement si les deux mots sont frères ou cousins.
- 4) Dans le programme "**PgmeFRCO**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterFRCO**" en exploitant l'**annexe** ci-après.
  - Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Vérifier**", permettant de récupérer les deux mots (**MotA** et **MotB**) saisis, puis d'exploiter les fonctions **Freres** et **Cousins** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterFRCO**".

## Exemples d'exécution

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

**Les mots frères-cousins**

Introduire MotA, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Introduire MotB, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Désolé ! vérifier la saisie des mots !!

**Les mots frères-cousins**

Introduire MotA, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Introduire MotB, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Désolé ! vérifier la saisie des mots !!

**Les mots frères-cousins**

Introduire MotA, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Introduire MotB, un mot non vide  
contenant que des lettres alphabétiques minuscules :

kamel est frère de riheb

**Les mots frères-cousins**

Introduire MotA, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Introduire MotB, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Le mot ali est cousin de bon

**Les mots frères-cousins**

Introduire MotA, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Introduire MotB, un mot non vide  
contenant que des lettres alphabétiques minuscules :

Les deux mots ni frères ni cousins

**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 <b>InterFRCO</b>	4 pts
Création et enregistrement du programme <b>PgmeFRCO</b>	1 pt
Développement des fonctions <b>Freres et Cousins</b>	6pts
Ajout des instructions: * De l'interface <b>InterFRCO</b> * Du module <b>Play</b>	2 pts 4 pts

- Important :
- Une solution modulaire au problème posé est exigée.
  - Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Une chaîne presque palindrome

- Une chaîne **presque palindrome**, est une chaîne **non vide** de nombre **pair** de caractères, pour laquelle les caractères correspondants de même rang à partir de la droite et de gauche sont de même famille deux à deux.
  - Deux caractères de même famille, s'ils ont les deux consonnes, les deux voyelles, les deux chiffres ou bien les deux symboles.
- NB :** la chaîne **presque palindrome** ne doit pas être **palindrome !!**

### Exemples :

① ch = "Tothem" est **presque palindrome**, car :

- Le 1<sup>er</sup> caractère "T" et le caractère correspondant de même rang à partir de la droite, 6<sup>ème</sup> caractère "m" sont les deux **consonnes**,
- Le 2<sup>ème</sup> caractère "o" et le caractère correspondant de même rang à partir de la droite, 5<sup>ème</sup> caractère "e" sont les deux **voyelles**,
- Le 3<sup>ème</sup> caractère "t" et le caractère correspondant de même rang à partir de la droite, 5<sup>ème</sup> caractère "h" sont les deux **consonnes**,

② ch = "B2#Ay!3B" est **presque palindrome**, car :

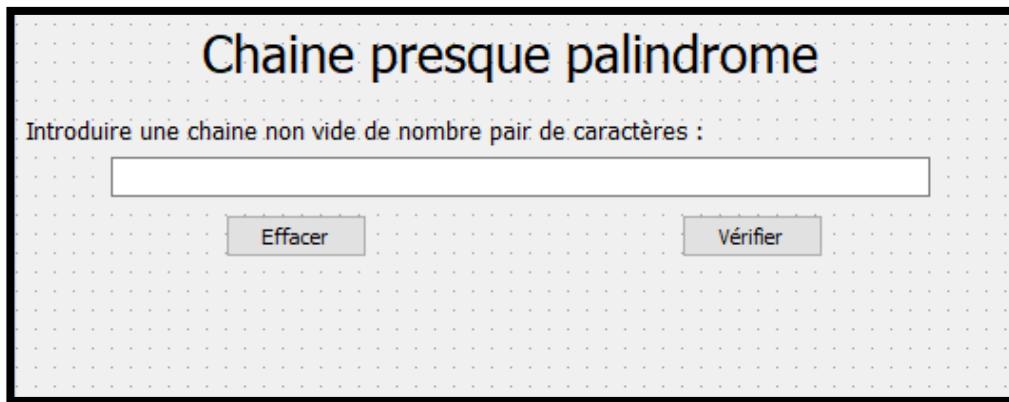
- Le 1<sup>er</sup> caractère "B" et le caractère correspondant de même rang à partir de la droite, 8<sup>ème</sup> caractère "B" sont les deux **consonnes**,
- Le 2<sup>ème</sup> caractère "2" et le caractère correspondant de même rang à partir de la droite, 7<sup>ème</sup> caractère "3" sont les deux **chiffres**,
- Le 3<sup>ème</sup> caractère "#" et le caractère correspondant de même rang à partir de la droite, 6<sup>ème</sup> caractère "!" sont les deux **symboles**,
- Le 4<sup>ème</sup> caractère "A" et le caractère correspondant de même rang à partir de la droite, 6<sup>ème</sup> caractère "y" sont les deux **voyelles**,

③ ch = "2#AC!3" n'est pas une chaîne **presque palindrome**, car :

- Le 1<sup>er</sup> caractère "2" et le caractère correspondant de même rang à partir de la droite, 6<sup>ème</sup> caractère "3" sont les deux **consonnes**,
- Le 2<sup>ème</sup> caractère "#" et le caractère correspondant de même rang à partir de la droite, 5<sup>ème</sup> caractère "!" sont les deux **symboles**,
- Le 3<sup>ème</sup> caractère "A" et le caractère correspondant de même rang à partir de la droite, 4<sup>ème</sup> caractère "C" sont les deux **de familles différentes (le 3<sup>ème</sup> est voyelle et le 4<sup>ème</sup> est consonne)**, donc ch n'est pas **presque palindrome**.

Pour vérifier si une chaîne **ch** non vide de nombre pair de caractères est **presque palindrome** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Chaine presque palindrome**"
- Un label demandant la saisie de la chaîne :  
"Introduire une chaîne non vide de nombre pair de caractères :"
- Une zone de saisie permettant la saisie de la chaîne.
- Un bouton intitulé " **Vérifier**"
- Un bouton intitulé " **Effacer**" (*permettant d'effacer la zone de saisie et le label d'affichage*)
- 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 "**InterPP**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmePP**".
- 3) Développer dans le programme "**PgmePP**", une fonction **Est\_PP(ch)** qui permet de vérifier si la chaîne **ch** est presque palindrome ou non.
- 4) Dans le programme "**PgmePP**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterPP**" en exploitant **l'annexe** ci-après.
  - Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Vérifier**", permettant de récupérer la chaîne **ch** saisie, puis d'exploiter la fonction **Est\_PP** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterPP**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

## Exemples d'exécution

### Chaine presque palindrome

Introduire une chaîne non vide de nombre pair de caractères :

Désolé ! vérifier la saisie de la chaîne !!

### Chaine presque palindrome

Introduire une chaîne non vide de nombre pair de caractères :

ELLE une chaîne n'est pas presque palindrome

### Chaine presque palindrome

Introduire une chaîne non vide de nombre pair de caractères :

La chaîne Tohem est presque palindrome

### Chaine presque palindrome

Introduire une chaîne non vide de nombre pair de caractères :

La chaîne B2#Ay!3B est presque palindrome

### Chaine presque palindrome

Introduire une chaîne non vide de nombre pair de caractères :

2#AC!3 une chaîne n'est pas presque palindrome

## 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 <b>InterPP</b>	4 pts
Création et enregistrement du programme <b>PgmePP</b>	1 pt
Développement de la fonction <b>Est_PP</b>	6pts
Ajout des instructions : * De l'interface <b>InterPP</b> * Du module <b>Play</b>	2 pts 4 pts
Importation des bibliothèques nécessaires, modularité et cohérence	3 pts

Lycée Béja Nord

# Sujet Pratique n°7

Durée : 1 heure

Classes : 4<sup>ème</sup> SC&M

Matière : Informatique

Enseignant : Kais TABOUBI

Date : \*\*-\*\*-2023

Nom et Prénom : .....

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre pointu

- Soit  $N$  un entier naturel strictement supérieur à 1. On dit que  $N$  est **pointu**, si  $N$  admet au plus un seul facteur premier ou bien si en notant  $p$  et  $q$  les deux plus grands facteurs premiers de  $N$ , avec  $p > q$ , l'inégalité  $p \geq 2 * q$  est vérifiée.

**Exemples :**

- ①  $N = 25$  est un nombre **pointu** car il n'a qu'un seul facteur premier.
- ②  $N = 147$  est un nombre **pointu** car  $147 = 3 * 7 * 7$ , les deux plus grands facteurs premiers de  $N$  avec  $p > q$  sont (7 et 3) et  $7 \geq 2 * 3$ .
- ③  $N = 105$  n'est pas un nombre **pointu** car  $105 = 3 * 5 * 7$ , les deux plus grands facteurs premiers de  $N$  avec  $p > q$  sont (7 et 5) et  $7 < 2 * 5$ .

Pour vérifier si un entier naturel  $N$  ( $N > 1$ ) est un nombre **pointu** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombre pointu**"
- Un label demandant la saisie du nombre :  
"Introduire un entier >1 :"
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- Un label pour afficher le message adéquat.

The diagram shows a user interface window titled "Nombre pointu". At the top, there is a label "Introduire un entier >1 :". Below it is a text input field. At the bottom, there are two buttons: "Effacer" on the left and "Vérifier" on the right. A message area is located at the bottom of the window.

## Travail demandé :

- 1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**InterPointu**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**NbrPointu**".
- 3) Développer dans le programme "**NbrPointu**", une fonction **Pointu (N)** qui permet de vérifier si l'entier **N** saisi est **pointu** ou non.
- 4) Dans le programme "**NbrPointu**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterPointu**" 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 **Pointu** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterPointu**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

**Nombre pointu**

Introduire un entier >1 :

**Effacer** **Vérifier**

Désolé ! vérifier la saisie du nombre !!

**Nombre pointu**

Introduire un entier >1 :

**Effacer** **Vérifier**

Désolé ! vérifier la saisie du nombre !!

**Nombre pointu**

Introduire un entier >1 :

**Effacer** **Vérifier**

Le nombre 91 n'est pas pointu

**Nombre pointu**

Introduire un entier >1 :

**Effacer** **Vérifier**

88 est un nombre pointu

**Nombre pointu**

Introduire un entier >1 :

**Effacer** **Vérifier**

Le nombre 75 n'est pas pointu

**Nombre pointu**

Introduire un entier >1 :

**Effacer** **Vérifier**

44 est un nombre pointu

## 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 :**

<b>Tâches</b>	<b>Nombre de points</b>
Conception de l'interface <b>InterPointu</b>	<b>4 pts</b>
Création et enregistrement du programme <b>NbrPointu</b>	<b>1 pt</b>
Développement de la fonction <b>Pointu</b>	<b>6 pts</b>
Ajout des instructions : * De l'interface <b>InterPointu</b> * Du module <b>Play</b>	<b>2 pts</b> <b>4 pts</b>
Importation des bibliothèques nécessaires, modularité et cohérence	<b>3 pts</b>

Lycée Béja Nord

# Sujet Pratique n°8

Durée : 1 heure

Classes : 4<sup>ème</sup> SC&M

Matière : Informatique

Enseignant : Kais TABOUBI

Date : \*\*-\*\*-2023

Nom et Prénom : .....

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre spécial

- Un nombre **N**, est un entier naturel de quatre chiffres **abcd** est dit **spécial**, si les quatre chiffres **a, b, c** et **d** sont non nuls et vérifient  $a^2 + b^2 + c^2 + d^2 = N^2$

**Exemples :**

- ① Le nombre **4879** est **spécial** car :  $4^2 + 8^2 + 7^2 + 9^2 = 10^2$
- ② Le nombre **3256** n'est pas **spécial** car :  $3^2 + 2^2 + 5^2 + 6^2 < 6^2$

Pour vérifier si un entier naturel de quatre chiffres **N** est un nombre **spécial** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombre spécial**"
- Un label demandant la saisie du nombre :  
"Introduire un entier de quatre chiffres non nuls :"
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- Un label pour afficher le message adéquat.

The diagram shows a window titled "Nombre spécial". Inside, there is a label "Introduire un entier de quatre chiffres non nuls :" followed by a text input field. Below the input field are two buttons: "Effacer" and "Vérifier".

## Travail demandé :

- 1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**InterSPEC**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeSPEC**".
- 3) Développer dans le programme "**PgmeSPEC**", une fonction **Special (N)** qui permet de vérifier si l'entier **N** saisi est **spécial** ou non.
- 4) Dans le programme "**PgmeSPEC**" :
  - ◎ Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterSPEC**" 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 **Special** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterSPEC**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

Exemples d'exécution

The figure consists of five separate windows, each titled "Nombre spécial". Each window contains the following elements:

- An input field labeled "Introduire un entier de quatre chiffres non nuls :".
- A grey "Effacer" button.
- A blue "Vérifier" button.
- A message area below the buttons.

The five windows show the following interactions:

- The first window has the input field containing "Blabla". The message area says "Désolé ! vérifier la saisie du nombre !!".
- The second window has the input field containing "523". The message area says "Désolé ! vérifier la saisie du nombre !!".
- The third window has the input field containing "3206". The message area says "Désolé ! vérifier la saisie du nombre !!".
- The fourth window has the input field containing "3967". The message area says "3967 est un nombre spécial".
- The fifth window has the input field containing "6423". The message area says "Le nombre 6423 n'est pas spécial".

## 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 <b>InterSPEC</b>	<b>4 pts</b>
Création et enregistrement du programme <b>PgmeSPEC</b>	<b>1 pt</b>
Développement de la fonction <b>Special</b>	<b>6 pts</b>
Ajout des instructions : * De l'interface <b>InterSPEC</b> * Du module <b>Play</b>	<b>2 pts</b> <b>4 pts</b>
Importation des bibliothèques nécessaires, modularité et cohérence	<b>3 pts</b>

**Important :**

Nom et Prénom : .....

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre totalement impair

Un nombre N, est dit **totalement impair**, si **tous ses chiffres impairs occupent des positions impaires**.

Les positions des chiffres sont numérotées de la droite vers la gauche en commençant par 1.

**Exemple :**

Nombre : 2 0 7 4 5

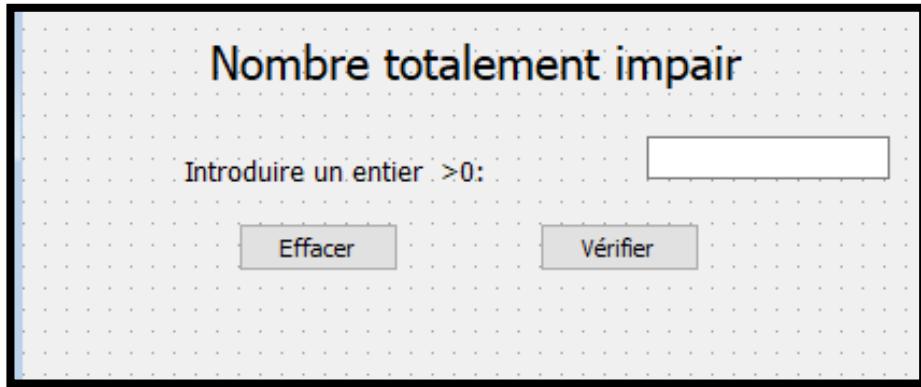
Position : 5 4 3 2 1

**Exemples :**

- ① N = 347 est un nombre **totalement impair**, car ses chiffres impairs sont 7 et 3 occupent respectivement les positions **1** et **3** qui sont impaires.
- ② N = 6745 est un nombre **totalement impair**, car ses chiffres impairs sont **5** et **7** occupent respectivement les positions **1** et **3** qui sont impaires.
- ③ N = 213 n'est pas un nombre **totalement impair**, car ses chiffres impairs sont **3** et **1** occupent respectivement les positions **1** et **2** alors que cette dernière (la position **2** pour le chiffre **1**) n'est pas impaire.
- ④ N = 5329 n'est pas un nombre **totalement impair**, car ses chiffres impairs sont **9**, **3** et **5** occupent respectivement les positions **1**, **3** et **4** alors que le dernier chiffre **5** occupe une position paire.

Pour vérifier si un entier naturel N strictement positif est un nombre **totalement impair** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombre totalement impair**"
- Un label demandant la saisie du nombre :  
**"Introduire un entier >0:"**
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "**Vérifier**"
- Un bouton intitulé "**Effacer**" (permettant d'effacer la zone de saisie et le label d'affichage)
- 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 "**InterTOTIMP**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeTOTIMP**".
- 3) Développer dans le programme "**PgmeTOTIMP**", une fonction **Tot\_impaire (N)** qui permet de vérifier si l'entier **N** saisi est **totalelement impair** ou non.
- 4) Dans le programme "**PgmeTOTIMP**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterTOTIMP**" 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 **Tot\_impaire** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterTOTIMP**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

## Exemples d'exécution

**Nombre totalement impair**

Introduire un entier >0:

Désolé ! vérifier la saisie du nombre !!

**Nombre totalement impair**

Introduire un entier >0:

Désolé ! vérifier la saisie du nombre !!

**Nombre totalement impair**

Introduire un entier >0:

2922 est un nombre totalement impair

**Nombre totalement impair**

Introduire un entier >0:

Le nombre 3967 n'est pas totalement impair

**Nombre totalement impair**

Introduire un entier >0:

509 est un nombre totalement impair

**Nombre totalement impair**

Introduire un entier >0:

Le nombre 253 n'est pas totalement impair

## 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 <b>InterTOTIMP</b>	<b>4 pts</b>
Création et enregistrement du programme <b>PgmeTOTIMP</b>	<b>1 pt</b>
Développement de la fonction <b>Tot_impar</b>	<b>6 pts</b>
Ajout des instructions : * De l'interface <b>InterTOTIMP</b>	<b>2 pts</b>
* Du module <b>Play</b>	<b>4 pts</b>
Importation des bibliothèques nécessaires, modularité et cohérence	<b>3 pts</b>

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre doublement palindrome

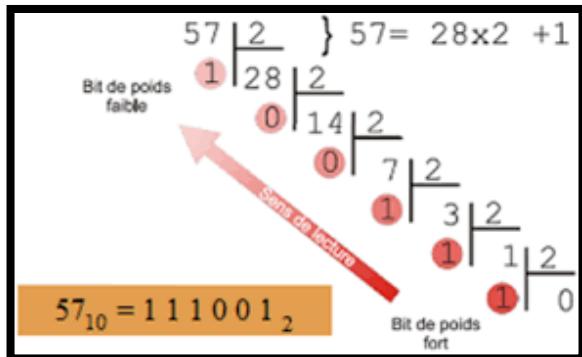
□ Un nombre naturel  $N$  ( $N \geq 1$ ), est dit **doublement palindrome** lorsque qu'il est :

- **palindrome** sous sa forme décimale
- et **palindrome** avec sa représentation **binaire**.

**Définitions :**

① **Un nombre palindrome:** est un nombre qui peut se lire indifféremment de gauche à droite ou de droite à gauche en gardant le même sens. (**Exemple:** 22, 313)

② **Un nombre binaire:** dans notre cas, est le résultat de la conversion d'un nombre décimal. Pour réaliser cette opération, il faut faire des divisions entières successives par 2 jusqu'à ce que le quotient devienne **nul**.

**Exemple :****Exemples :**

$N=313$  est **un nombre doublement palindrome** car :

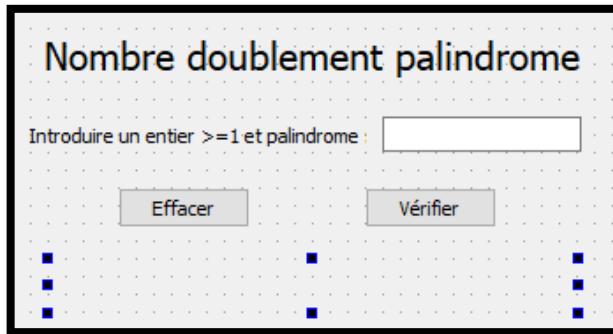
1. il est palindrome avec sa représentation décimale (**313**)
2. et palindrome avec sa représentation **binaire** (**313** en **binaire** est égal à **100111001**)

$N=22$  n'est pas **un nombre doublement palindrome** car :

1. il est palindrome avec sa représentation décimale (**22**)
2. mais n'est pas palindrome avec sa représentation **binaire** (**22** en **binaire** est égal à **10110**)

Pour vérifier si un entier naturel  $N$  ( $N \geq 1$ ) palindrome, est un nombre **doublement palindrome** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombré doublement palindrome**"
- Un label demandant la saisie du nombre : "**Introduire un entier  $\geq 1$  et palindrome :**"
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (*permettant d'effacer la zone de saisie et le label d'affichage*)
- 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 "**InterDP**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**NbrDP**".
- 3) Développer dans le programme "**NbrDP**", une fonction **DP (N)** qui permet de vérifier si l'entier **N** saisi est **doublement palindrome** ou non.
- 4) Dans le programme "**NbrDP**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterDP**" 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 **DP** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterDP**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

## Exemples d'exécution

<p><b>Nombre doublement palindrome</b></p> <p>Introduire un entier <math>\geq 1</math> et palindrome : <input type="text" value="blabla"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Désolé! Introduire un nombre valide !!!</p>	<p><b>Nombre doublement palindrome</b></p> <p>Introduire un entier <math>\geq 1</math> et palindrome : <input type="text" value="0"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Désolé! Introduire un nombre valide !!!</p>
<p><b>Nombre doublement palindrome</b></p> <p>Introduire un entier <math>\geq 1</math> et palindrome : <input type="text" value="20"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Désolé! Introduire un nombre valide !!!</p>	<p><b>Nombre doublement palindrome</b></p> <p>Introduire un entier <math>\geq 1</math> et palindrome : <input type="text" value="99"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Le nombre 99 est doublement palindrome</p>
<p><b>Nombre doublement palindrome</b></p> <p>Introduire un entier <math>\geq 1</math> et palindrome : <input type="text" value="161"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>161 n'est pas un nombre doublement palindrome</p>	<p><b>Nombre doublement palindrome</b></p> <p>Introduire un entier <math>\geq 1</math> et palindrome : <input type="text" value="585"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Le nombre 585 est doublement palindrome</p>

**Important :****Nom et Prénom :** .....

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre de Smith

- **Un nombre de Smith N**, est un entier naturel strictement supérieur à 1, dont la somme des chiffres est la même que celle des chiffres de ses facteurs premiers, incluant la répétitions de facteurs.

### Exemples :

- ① N= **636** est **un nombre de Smith** car: ses facteurs sont **2\*2\*3\*53**. La somme des chiffres de **636** est **15 (6+3+6)** et celle des chiffres des facteurs est également **15 (2+2+3+5+3)**
- ② N= **27** est **un nombre de Smith** car: ses facteurs sont **3\*3\*3**. La somme des chiffres de **27** est **9 (2+7)** et celle des chiffres des facteurs est également **9 (3+3+3)**
- ③ N= **25** n'est pas **un nombre de Smith** car: ses facteurs sont **5\*5**. La somme des chiffres de **25** est **7 (2+5)** et celle des chiffres des facteurs est également **10 (5+5)**
- ④ N= **56** n'est pas **un nombre de Smith** car: ses facteurs sont **2\*2\*2\*7**. La somme des chiffres de **56** est **11 (5+6)** et celle des chiffres des facteurs est également **13 (2+2+2+7)**

Pour vérifier si un entier naturel **N** (**N >1**) est un nombre **de Smith** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombre de Smith**"
- Un label demandant la saisie du nombre :  
"**Introduire un entier > 1 :**"
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- 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 "**InterSmith**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**NbrSmith**".
- 3) Développer dans le programme "**NbrSmith**", une fonction **Smith** (**N**) qui permet de vérifier si l'entier **N** saisi est **Smith** ou non.
- 4) Dans le programme "**NbrSmith**" :
  - ◎ Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterSmith**" 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 **Smith** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterSmith**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

Exemples d'exécution

Nombre de Smith

Introduire un entier >1 :

Désolé ! vérifier la saisie du nombre !!

Nombre de Smith

Introduire un entier >1 :

Désolé ! vérifier la saisie du nombre !!

Nombre de Smith

Introduire un entier >1 :

Le nombre 56 n'est pas Smith

Nombre de Smith

Introduire un entier >1 :

666 est un nombre de Smith

Nombre de Smith

Introduire un entier >1 :

Le nombre 525 n'est pas Smith

Nombre de Smith

Introduire un entier >1 :

706 est un nombre de Smith

## 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 <b>InterSmith</b>	<b>4 pts</b>
Création et enregistrement du programme <b>NbrSmith</b>	<b>1 pt</b>
Développement de la fonction <b>Smith</b>	<b>6 pts</b>
Ajout des instructions : * De l'interface <b>InterSmith</b> * Du module <b>Play</b>	<b>2 pts</b> <b>4 pts</b>
Importation des bibliothèques nécessaires, modularité et cohérence	<b>3 pts</b>

Important : Nom et Prénom : .....

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Détermination des nombres SECTIONNABLE UNITAIRES

On dit qu'un nombre entier est **sectionnable unitaire** s'il est supérieur ou égal à 3 et s'il peut s'écrire sous la forme :  $1 + 2 + 3 + \dots + p$  où  $p$  est un entier supérieur ou égal à 2.

### Exemple 1:

3 est un nombre **Sectionnable unitaire**.

$$\text{Car: } 3 = 1 + 2$$

### Exemple 2:

10 est un nombre **Sectionnable unitaire**.

$$\text{Car: } 10 = 1 + 2 + 3 + 4$$

### Exemple 3:

16 n'est pas un nombre **Sectionnable unitaire**.

$$\text{Car: } 16 > 1 + 2 + 3 + 4 + 5$$

$$16 < 1 + 2 + 3 + 4 + 5 + 6$$

### Exemple 4:

21 est un nombre **Sectionnable unitaire**.

$$\text{Car: } 21 = 1 + 2 + 3 + 4 + 5 + 6$$

### Exemple 5:

136 est un nombre **Sectionnable unitaire**.

$$\text{Car: } 136 = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16$$

## Travail demandé :

Ecrire un programme en **Python** qui permet de :

- Remplir un tableau T, par m entiers naturels supérieurs ou égal à 3, avec ( $4 \leq m \leq 20$ ).
- Déterminer et afficher, à partir du tableau T,
  - (C) Les entiers **Sectionnable unitaires** (sans redondance) selon la description décrite précédemment, s'il existe,
  - (C) Sinon le message suivant "**Aucun nombre Sectionnable unitaire dans le tableau**"

Exemple 1 : Soit  $m=7$  et le tableau T suivant :

T	11	10	15	16	136	67	15
	0	1	2	3	4	5	6

le programme affiche : **les nombres sectionnable unitaires sont : 10, 15, 136**

Exemple 2 : Soit  $m=4$  et le tableau T suivant :

T	23	11	103	386
	0	1	2	3

le programme affiche : **Aucun nombre sectionnable unitaire dans le tableau**

Important :

Nom et Prénom : .....

- Une solution modulaire au problème posé est exigée.
- Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Détermination des nombres ETRANCHES

Un entier naturel **N**, composé de **p** chiffres ( $N = n_1n_2\dots n_p$ ) est un nombre **ETRANCHE** si, pour tout entier **i** allant de **2** à **p**, l'entier naturel  $n_1n_2\dots n_i$  est divisible par **i**.

**Exemple 1:** 2016 est un nombre **ETRANCHE**.

En effet :

- Ⓐ 20 est divisible par 2
- Ⓐ 201 est divisible par 3
- Ⓐ 2016 est divisible par 4

**Exemple 2:** 201920 n'est pas un nombre **ETRANCHE**.

Car :

- Ⓐ 20 est divisible par 2
- Ⓐ 201 est divisible par 3
- Ⓐ 2019 n'est pas divisible par 4, **donc on arrête la vérification**.

**Exemple 3:** 705 est un nombre **ETRANCHE**.

En effet :

- Ⓐ 70 est divisible par 2
- Ⓐ 705 est divisible par 3

**Exemple 4:** 72 est un nombre **ETRANCHE**.

En effet :

- Ⓐ 72 est divisible par 2

**Exemple 5:** 2041 n'est pas un nombre **ETRANCHE**. Car :

- Ⓐ 20 est divisible par 2
- Ⓐ 204 est divisible par 3
- Ⓐ 2041 n'est pas divisible par 4, **donc on arrête la vérification**.

### Travail demandé :

Ecrire un programme en **Python** qui permet de :

- Remplir un tableau **T**, par **m** entiers naturels de deux chiffres ou plus, avec ( $4 \leq m \leq 20$ ).
- Déterminer et afficher, à partir du tableau **T**,
  - Ⓐ Les entiers **ETRANCHES** selon la description décrite précédemment, s'il existe,
  - Ⓐ Sinon le message suivant "**Aucun nombre ETRANCHE dans le tableau**"

**Exemple1 :** Soit **m=7** et le tableau **T** suivant :

<b>T</b>	23	102	72	705	705204	1024	326
	0	1	2	3	4	5	6

le programme affiche : **les nombres ETRANCHES sont : 102, 72, 705, 705204, 1024**

**Exemple2 :** Soit **m=4** et le tableau **T** suivant :

<b>T</b>	23	326	103	705234
	0	1	2	3

le programme affiche : **Aucun nombre ETRANCHE dans le tableau**

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

On dispose d'un tableau **T** contenant **N** entiers **strictement positifs palindromes**.

L'algorithme suivant est celui d'une fonction permettant de vérifier si un entier **X** est palindrome ou non.

**Fonction Palindrome (X : entier) : booléen**

**Début**

ch ← **convch(X)**

i ← 0

**Tant que** i < **Long(ch)** **Div 2 ET** ch[i] = ch[**Long(ch)** -1 - i] **Faire**

i ← i +1

**Fin Tant que**

**Retourner** i = **Long(ch)** **Div 2**

**Fin**

TDOL	
Objet	Type/Nature
i	Entier
ch	Chaine

**Questions :**

Utiliser la fonction ci-dessus, pour écrire un programme en Python, permettant de :

- Saisir un tableau **T** de **N** entiers strictement positifs palindromes (**5 ≤ N ≤ 20**).
- Afficher, s'il existe, les nombres doublement palindromes. Comme le montre l'exemple ci-dessous.

Un nombre naturel **N** (**N ≥ 1**), est dit **doublement palindrome** lorsque qu'il est :

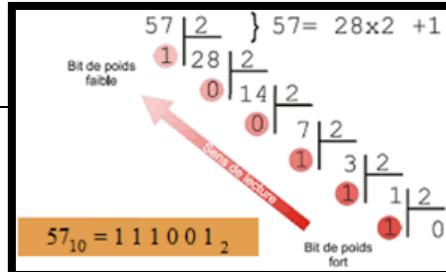
- **palindrome** sous sa forme décimale
- et **palindrome** avec sa représentation **binaire**.

**Définitions :**

**① Un nombre palindrome:** est un nombre qui peut se lire indifféremment de gauche à droite ou de droite à gauche en gardant le même sens. (**Exemple:** 22, 313)

**② Un nombre binaire:** dans notre cas, est le résultat de la conversion d'un nombre décimal.

Pour réaliser cette opération, il faut faire des divisions entières successives par **2** jusqu'à ce que le quotient devienne **nul**. **Exemple :**



**Exemple :** Pour le tableau T suivant avec N=8 :

T	11	4	33	212	414	717	22	99
	0	1	2	3	4	5	6	7

Le programme affiche :

*Les nombres doublement palindromes sont :*

**33** = 100001

**717**= 1011001101

**99**=1100011

Important : Nom et Prénom : .....

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Concours d'une commune

Une commune organise sur un week-end un concours de **course / marche / randonnée** par équipes.

Pour gérer les résultats de la compétition, les organisateurs ont créé deux tableaux :

- Un tableau **TN** (Tableau des Noms) dans lequel sont enregistrés les noms des équipes.
- Un tableau **TT** (Tableau de Temps) dans lequel est enregistré le temps total pris une équipe pour réaliser les épreuves. Le temps est mesuré en secondes.

A un nom inscrit dans la case d'indice **i** de **TN** correspond le temps inscrit dans la case d'indice **i** de **TT**.

Ainsi par exemple, avec **4** équipes, on peut avoir :

TN	Les bisounours	La clef des champs	Vert de gris	Les schtroumpfs
	1	2	3	4
TT	7357	7045	9283	7045
	1	2	3	4

Les deux équipes " **La clef des champs** " et " **Les schtroumpfs** " sont pour le moment en tête du concours avec un temps total de **7045** secondes, talonnées par l'équipe " **Les bisounours** ", en troisième position, avec **7357** secondes. Enfin de peloton, arrive l'équipe " **Vert de gris** " avec un temps de **9283** secondes.

On se propose de réaliser un programme en Python intitulé "**COMPETITION**" qui permet de :

- Remplir, à la fois, deux tableaux **TN** et **TT** par respectivement, **N** noms d'équipes avec (**3 ≤ N ≤ 20**) et le temps total pris chaque équipe pour réaliser les épreuves.

**NB :** Chaque nom d'équipe est un ensemble de mots alphabétiques distincts bien espacés

- Afficher le classement final du concours comme indiqué dans l'exemple ci-après, en commençant par le nom de l'équipe qui a réalisé le plus petit temps. Il est à noter que les équipes ayant un même temps auront un même rang dans le classement.

**NB :** chaque ligne d'affichage doit contenir, suite au nom de l'équipe, le temps total sous la forme en **H:Min:Sec**. Par exemple, le temps pris par l'équipe " **Les bisounours** " est **7357 secondes**, lors de l'affichage final sera :

**Rang n° 3 :** L'équipe **Les bisounours** a pris **2 H :3 Min :37 Sec** pour réaliser toutes les épreuves

**Exemple :** pour les données enregistrées dans les deux tableaux **TN** et **TT** citées précédemment,

*Le programme affichera*

Le classement est :

Rang n°1 : L'équipe **La clef des champs** a pris **1 H :57 Min :25 Sec** pour réaliser toutes les épreuves

L'équipe **Les schtroumpfs** a pris **1 H :57 Min :25 Sec** pour réaliser toutes les épreuves

Rang n°3 : L'équipe **Les bisounours** a pris **2 H :3 Min :37 Sec** pour réaliser toutes les épreuves

Rang n°4 : L'équipe **Vert de gris** a pris **2 H :34 Min :43 Sec** pour réaliser toutes les épreuves

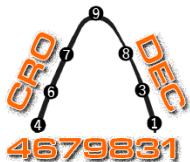
**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre CRO-DEC et le nombre DEC-CRO

- Un nombre naturel **N** ( $N \geq 100$ ), est dit **CRO-DEC** (c-à-d **CROISSANT** ensuite **DECROISSANT**) s'il est formé d'un **nombre impair de chiffres** et vérifie les propriétés suivantes :

- Les chiffres de la partie gauche du nombre **N** doivent être en **ordre croissant stricte**,
- les chiffres de la partie droite du nombre doivent être en **ordre décroissant stricte**,
- et le chiffre du milieu sera le **plus grand des chiffres de gauche et de droite**.



Par exemple : **N** s'écrit sous la forme  $C_1C_2C_3C_4C_5$ . Les  $C_i$ , doivent se présenter de la manière

suivante :  $C_1 < C_2 < C_3 > C_4 > C_5$ . Avec  $i \in \{1, \dots, 5\}$  et  $C_i$  sont les 5 chiffres qui le composent.

- Un nombre naturel **N** ( $N \geq 100$ ), est dit **DEC-CRO** (c-à-d **DECROISSANT** ensuite **CROISSANT**) s'il est formé d'un **nombre impair de chiffres** et vérifie les propriétés suivantes :

- Les chiffres de la partie gauche du nombre **N** doivent être en **ordre décroissant stricte**,
- les chiffres de la partie droite du nombre doivent être en **ordre croissant stricte**,
- et le chiffre du milieu sera le **plus petit des chiffres de gauche et de droite**.



Par exemple : **N** s'écrit sous la forme  $C_1C_2C_3C_4C_5$ . Les  $C_i$ , doivent se présenter de la manière

suivante :  $C_1 > C_2 > C_3 < C_4 < C_5$ . Avec  $i \in \{1, \dots, 5\}$  et  $C_i$  sont les 5 chiffres qui le composent.

### Exemples :

#### Des nombres CRO-DEC

**N= 283** : N est un nombre **CRO-DEC**, car  $2 < 8$  et  $8 > 3$

**N= 35741** : N est un nombre **CRO-DEC**, car  $3 < 5 < 7 > 4 > 1$

#### Des nombres DEC- CRO

**N= 724** : N est un nombre **DEC-CRO**, car  $7 > 2 < 4$

**N= 85169** : N est un nombre **DEC-CRO**, car  $8 > 5 > 1 < 6 < 9$

#### Des nombres ni CRO-DEC ni DEC- CRO

**N= 289** : N n'est pas un nombre **CRO-DEC**, car  $2 < 8$  mais  $8 < 9$   
Et N n'est pas un nombre **DEC-CRO**, car  $2 < 8$

**N= 35541** : N n'est pas un nombre **CRO-DEC**, car  $3 < 5$  mais le chiffre du milieu (5) n'est pas supérieur stricte au chiffre qui le précède et N n'est pas un nombre **DEC-CRO**, car  $3 < 5$

**N= 35749** : N n'est pas un nombre **CRO-DEC**, car  $3 < 5 < 7 > 4$  mais le dernier chiffre (9) n'est pas inférieur stricte au chiffre qui le précède et N n'est pas un nombre **DEC-CRO**, car  $3 < 5$

Pour vérifier si un entier naturel **N** ( $N \geq 100$ ) est un nombre **CRO-DEC**, **DEC-CRO** ou ni **CRO-DEC** ni **DEC-CRO**, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**CRO-DEC \*\*\* DEC-CRO**"

- Un label demandant la saisie du nombre :

"Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair: "

- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- Un label pour afficher le message adéquat

**CRO-DEC \*\* DEC-CRO**

Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair :

Effacer      Vérifier

## Travail demandé :

- 1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**InterCRODEC**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**NbrCRODEC**".
- 3) Développer dans le programme **NbrCRODEC**, deux fonctions **CRODEC(N)** et **DECRO(N)** qui permettent respectivement de vérifier si un entier N :
  - est un **nombre CRO-DEC**
  - est un **nombre DEC-CRO**
- 4) 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 les deux fonctions, afin d'afficher le message adéquat via le label dédié à l'affichage de l'interface "**InterCRODEC**"

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

**CRO-DEC \*\* DEC-CRO**

Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair :

Blabla

Effacer      Vérifier

Veuillez introduire un nombre  $\geq 100$  dont le nombre de ses chiffres est impair

**CRO-DEC \*\* DEC-CRO**

Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair :

99

Effacer      Vérifier

Veuillez introduire un nombre  $\geq 100$  dont le nombre de ses chiffres est impair

**CRO-DEC \*\* DEC-CRO**

Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair :

456325

Effacer      Vérifier

Veuillez introduire un nombre  $\geq 100$  dont le nombre de ses chiffres est impair

**CRO-DEC \*\* DEC-CRO**

Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair :

2469431

Effacer      Vérifier

2469431 est un nombre CRO-DEC

**CRO-DEC \*\* DEC-CRO**

Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair :

75368

Effacer      Vérifier

75368 est un nombre DEC-CRO

**CRO-DEC \*\* DEC-CRO**

Introduire un entier  $\geq 100$  et dont le nombre de ses chiffres est impair :

9456310

Effacer      Vérifier

9456310 ni CRO-DEC ni DEC-CRO

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

**Le mot emprisonné**

■ Un mot **MotA** est prisonnier (inclus) dans un autre mot **MotB**, si toutes les lettres du **MotA** se trouvent dans **MotB** dans **le même ordre**.

**NB :** \* Les deux mots ne doivent pas des chaînes **vides** et contiennent que des **lettres alphabétiques majuscules**.  
 \*\* La longueur du **MotA** est inférieure ou égale à celle du **MotB**.

**Exemples :**

① **MotA** = "CHAT" est prisonnier dans **MotB**="ATTACHANT" car:

- La 1<sup>ère</sup> lettre du **MotA** :"C" existe dans **MotB**,
- la 2<sup>ème</sup> lettre du **MotA** :"H" existe après la première position de la lettre "C" dans **MotB** (c'est-à-dire dans la sous chaîne "HANT"),
- la 3<sup>ème</sup> lettre du **MotA** :"A" existe après la première position de la lettre "H" dans **MotB** (c'est-à-dire dans la sous chaîne "ANT"),
- et la 4<sup>ème</sup> lettre du **MotA** :"T" existe après la première position de la lettre "A" dans **MotB** (c'est-à-dire dans la sous chaîne "NT").

② **MotA** = "LIGNE" n'est pas prisonnier dans **MotB**="SIGNALER" malgré que toutes les lettres de **MotA** se trouvent dans **MotB** car :

- La 1<sup>ère</sup> lettre du **MotA** :"L" existe dans **MotB**,
- **Mais** la 2<sup>ème</sup> lettre du **MotA** :"I" existe dans **MotB** et pas après la première position de la lettre "L" dans **MotB** (c'est-à-dire dans la sous chaîne "ER"), **donc** on arrête la vérification de l'emprisonnement du **MotA** dans **MotB**.

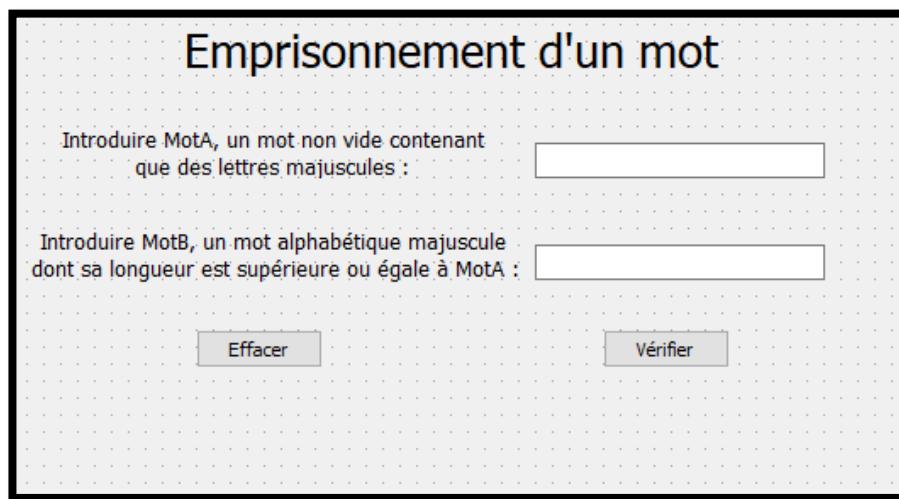
③ **MotA** = "TIGE" est prisonnier dans **MotB**="DISTINGUE" car:

- La 1<sup>ère</sup> lettre du **MotA** :"T" existe dans **MotB**,
- la 2<sup>ème</sup> lettre du **MotA** :"I" existe après la première position de la lettre "T" dans **MotB** (c'est-à-dire dans la sous chaîne "INGUE"),
- la 3<sup>ème</sup> lettre du **MotA** :"G" existe après la première position de la lettre "I" dans **MotB** (c'est-à-dire dans la sous chaîne "NGUE"),
- et la 4<sup>ème</sup> lettre du **MotA** :"E" existe après la première position de la lettre "G" dans **MotB** (c'est-à-dire dans la sous chaîne "UE"),

④ **MotA** = "CAGE" n'est pas prisonnier dans **MotB**="AVANTAGEUX" car : la 1<sup>ère</sup> lettre du **MotA** :"C" n'existe pas dans **MotB**, **donc** on arrête la vérification de l'emprisonnement du **MotA** dans **MotB**.

Pour vérifier si un mot **MotA** est prisonnier ou non dans un autre mot **MotB**, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Emprisonnement d'un mot**"
- Un label demandant la saisie du premier mot :  
"Introduire MotA, un mot non vide contenant que des lettres majuscules : "
- Une zone de saisie permettant la saisie du **MotA**.
- Un label demandant la saisie du second mot :  
"Introduire MotB, un mot alphabétique majuscule dont sa longueur est supérieure ou égale à MotA : "
- Une zone de saisie permettant la saisie du **MotB**.
- Un bouton intitulé " **Vérifier**"
- Un bouton intitulé " **Effacer**" (*permettant d'effacer la zone de saisie et le label d'affichage*)
- 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 "**InterEMPRIS**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**MotEmpris**".
- 3) Développer dans le programme "**MotEmpris**", une fonction **Emprisonnier (MotA, MotB)** qui permet de vérifier si **MotA** est prisonnier dans **MotB** ou non.
- 4) Dans le programme "**MotEmpris**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterEMPRIS**" en exploitant **l'annexe** ci-après.
  - Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Vérifier**", permettant de récupérer les deux mots (**MotA** et **MotB**) saisis, puis d'exploiter la fonction **Emprisonnier** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterEMPRIS**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

**Emprisonnement d'un mot**

Introduire MotA, un mot non vide contenant que des lettres majuscules :

Introduire MotB, un mot alphabétique majuscule dont sa longueur est supérieure ou égale à MotA :

Désolé ! vérifier la saisie des mots !!

**Emprisonnement d'un mot**

Introduire MotA, un mot non vide contenant que des lettres majuscules :

Introduire MotB, un mot alphabétique majuscule dont sa longueur est supérieure ou égale à MotA :

Désolé ! vérifier la saisie des mots !!

**Emprisonnement d'un mot**

Introduire MotA, un mot non vide contenant que des lettres majuscules :

Introduire MotB, un mot alphabétique majuscule dont sa longueur est supérieure ou égale à MotA :

TIGE est prisonnié dans le mot DISTINGUE

**Emprisonnement d'un mot**

Introduire MotA, un mot non vide contenant que des lettres majuscules :

Introduire MotB, un mot alphabétique majuscule dont sa longueur est supérieure ou égale à MotA :

Le mot LIGNE n'est pas prisonnié dans le mot SIGNALER

**Emprisonnement d'un mot**

Introduire MotA, un mot non vide contenant que des lettres majuscules :

Introduire MotB, un mot alphabétique majuscule dont sa longueur est supérieure ou égale à MotA :

Le mot LUNE n'est pas prisonnié dans le mot FORTUNE

**Emprisonnement d'un mot**

Introduire MotA, un mot non vide contenant que des lettres majuscules :

Introduire MotB, un mot alphabétique majuscule dont sa longueur est supérieure ou égale à MotA :

CHAT est prisonnié dans le mot ATTACHANT

## 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_()

```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Les nombres homogènes

□ Deux nombres **A** et **B** sont dites **homogènes** s'ils ont les mêmes facteurs premiers (sans tenir compte de leurs exposants).

NB : \* A et B : deux entiers naturels strictement supérieurs à **1**,  
\*\* A est différent de B.

**Exemples :**

① A=60 et B = 90 sont **homogènes**, leurs facteurs premiers sont mutuellement **2, 3 et 5**. En effet,

$$60 = 2^2 * 3 * 5 \quad \text{et} \quad 90 = 2 * 3^2 * 5$$

② A=63 et B = 84 ne sont pas **homogènes**, car leurs facteurs premiers **distincts** ne sont pas les mêmes. En effet,

$$63 = 3^2 * 7 : \text{leurs facteurs } \textit{distincts}, \text{ sans tenir compte de leurs exposants, sont (3 et 7)}$$

$$84 = 2^2 * 3 * 7 : \text{leurs facteurs } \textit{distincts}, \text{ sans tenir compte de leurs exposants, sont (2, 3 et 7)}$$

③ A=48 et B = 72 sont **homogènes**, leurs facteurs premiers sont mutuellement **2 et 3**. En effet,

$$48 = 2^4 * 3 \quad \text{et} \quad 72 = 2^3 * 3^2$$

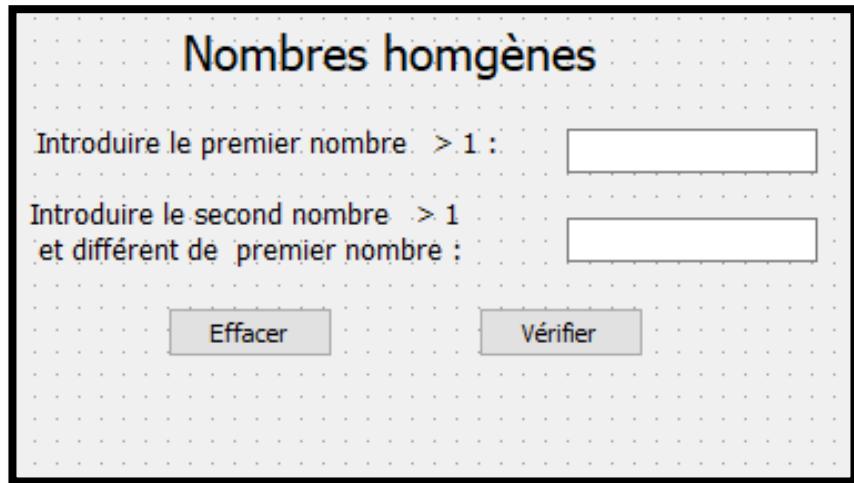
④ A=50 et B = 70 ne sont pas **homogènes**, car leurs facteurs premiers **distincts** ne sont pas les mêmes. En effet,

$$50 = 2 * 5^2 : \text{leurs facteurs } \textit{distincts}, \text{ sans tenir compte de leurs exposants, sont (2 et 5)}$$

$$70 = 2 * 5 * 7 : \text{leurs facteurs } \textit{distincts}, \text{ sans tenir compte de leurs exposants, sont (2, 5 et 7).}$$

Pour vérifier si deux entiers **A** et **B** sont **homogènes** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "Les nombres homogènes"
- Un label demandant la saisie du premier nombre :  
"Introduire le premier nombre > 1 :"
- Une zone de saisie permettant la saisie du premier nombre.
- Un label demandant la saisie du second nombre :  
"Introduire le second nombre > 1 et différent de premier nombre :"
- Une zone de saisie permettant la saisie du second nombre.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- 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 "**InterHOMO**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeHOMO**".
- 3) Développer dans le programme "**PgmeHOMO**", une fonction **Homogene (A, B)** qui permet de vérifier si les deux entiers **A** et **B** saisis sont **homogènes** ou non.
- 4) Dans le programme "**PgmeHOMO**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterHOMO**" 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 **Homogene** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterHOMO**".

# Exemples d'exécution

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

<p><b>Nombres homogènes</b></p> <p>Introduire le premier nombre &gt; 1 : <input type="text" value="Blabla"/></p> <p>Introduire le second nombre &gt; 1 et différent de premier nombre : <input type="text" value="25"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Désolé ! vérifier la saisie des nombres !!</p>	<p><b>Nombres homogènes</b></p> <p>Introduire le premier nombre &gt; 1 : <input type="text" value="33"/></p> <p>Introduire le second nombre &gt; 1 et différent de premier nombre : <input type="text" value="33"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Désolé ! vérifier la saisie des nombres !!</p>
<p><b>Nombres homogènes</b></p> <p>Introduire le premier nombre &gt; 1 : <input type="text" value="25"/></p> <p>Introduire le second nombre &gt; 1 et différent de premier nombre : <input type="text" value="62"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Les deux nombres 25 et 62 ne sont pas homogènes</p>	<p><b>Nombres homogènes</b></p> <p>Introduire le premier nombre &gt; 1 : <input type="text" value="22"/></p> <p>Introduire le second nombre &gt; 1 et différent de premier nombre : <input type="text" value="88"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>22 et 88 sont homogènes</p>
<p><b>Nombres homogènes</b></p> <p>Introduire le premier nombre &gt; 1 : <input type="text" value="11"/></p> <p>Introduire le second nombre &gt; 1 et différent de premier nombre : <input type="text" value="88"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>Les deux nombres 11 et 88 ne sont pas homogènes</p>	<p><b>Nombres homogènes</b></p> <p>Introduire le premier nombre &gt; 1 : <input type="text" value="72"/></p> <p>Introduire le second nombre &gt; 1 et différent de premier nombre : <input type="text" value="48"/></p> <p><input type="button" value="Effacer"/> <input type="button" value="Vérifier"/></p> <p>72 et 48 sont homogènes</p>

## 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_()
```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le plus petit multiple binaire d'un nombre décimal

□ Pour se divertir pendant son temps libre, deux enfants s'amusent à inventer des codes. Le premier enfant propose, à son ami, de coder un entier naturel **N** par **son plus petit multiple non nul** contenant uniquement des zéros et des uns.

**NB:**

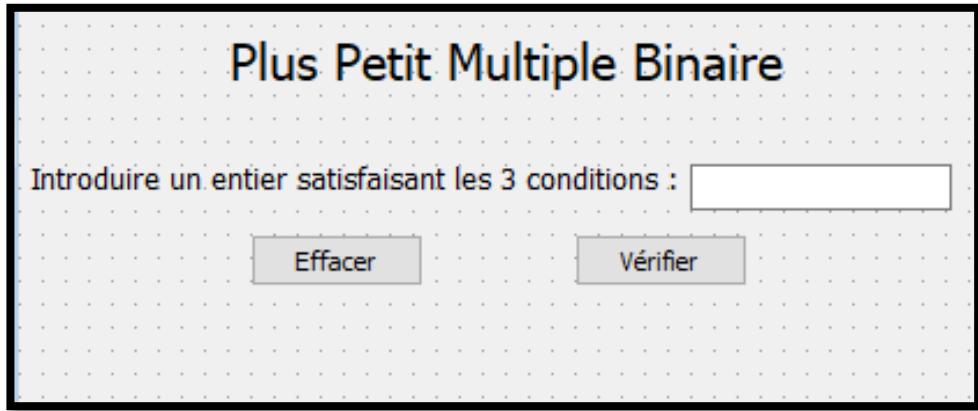
- \* L'entier **N** doit être dans l'intervalle **[1, 88]**,
- \*\* **N** n'est pas un multiple de **9**,
- \*\*\* **N** ne contient pas le chiffre **9**.

**Exemples :**

- ① Le plus petit multiple de **2** satisfaisant la condition est **10**. Donc le nombre **2** est codé en **10**
- ② Le plus petit multiple de **3** dont ses chiffres sont uniquement des zéros et (ou) des uns est **111**. (**3 \* 37 = 111**).
- ③ Le plus petit multiple de **4** dont ses chiffres sont uniquement des zéros et (ou) des uns est **100**. (**4 \* 25 = 100**).
- ④ Le plus petit multiple de **5** dont ses chiffres sont uniquement des zéros et (ou) des uns est **10**. (**5 \* 2 = 10**).
- ⑤ Le plus petit multiple de **6** dont ses chiffres sont uniquement des zéros et (ou) des uns est **1110**. (**6 \* 185 = 1110**).
- ⑥ Le plus petit multiple de **58** dont ses chiffres sont uniquement des zéros et (ou) des uns est **11011010**.  
**(58 \* 189845 = 11011010)**.

Pour déterminer **le plus petit multiple** d'un entier **N** dont chacun de ses chiffres est **zéro** ou **un**, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Plus Petit Multiple Binaire**"
- Un label demandant la saisie du nombre :  
"**Introduire un entier satisfaisant les 3 conditions :**"
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "**Vérifier**"
- Un bouton intitulé "**Effacer**" (*permettant d'effacer la zone de saisie et le label d'affichage*)
- 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 "**InterPPMB**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmePPMB**".
- 3) Développer dans le programme "**PgmePPMB**", une fonction **PPMB (N)** qui permet de déterminer et retourner **le plus petit multiple binaire** de l'entier **N**.
- 4) Dans le programme "**PgmePPMB**" :
  - Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterPPMB**" 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 satisfaisant les 3 conditions décrites précédemment, puis d'exploiter la fonction **PPMB** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterPPMB**".

## Exemples d'exécution

NB : L'affichage du message doit être conforme aux exemples d'exécution suivants :

Plus Petit Multiple Binaire

Introduire un entier satisfaisant les 3 conditions :

Désolé ! vérifier la saisie du nombre !!

Plus Petit Multiple Binaire

Introduire un entier satisfaisant les 3 conditions :

Désolé ! vérifier la saisie du nombre !!

Plus Petit Multiple Binaire

Introduire un entier satisfaisant les 3 conditions :

Désolé ! vérifier la saisie du nombre !!

Plus Petit Multiple Binaire

Introduire un entier satisfaisant les 3 conditions :

$78 * 1295 = 101010$

Plus Petit Multiple Binaire

Introduire un entier satisfaisant les 3 conditions :

$86 * 128035 = 11011010$

Plus Petit Multiple Binaire

Introduire un entier satisfaisant les 3 conditions :

$11 * 1 = 11$

## 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_()
```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre multipotent

- Un nombre multipotent N**, est un entier naturel non nul dont le produit des diviseurs est égal à une puissance de N.

### Exemples :

- ① Le nombre **30** est **multipotent** car :

Le produit de ses diviseurs ( $1 * 2 * 3 * 5 * 6 * 10 * 15 * 30 = 810\ 000$ ) est égal à  $30^4 = 810\ 000$ .

- ② Le nombre **6** est **multipotent** car :

Le produit de ses diviseurs ( $1 * 2 * 3 * 6 = 36$ ) est égal à  $6^2 = 36$ .

- ③ Le nombre **9** n'est pas **multipotent** car :

Le produit de ses diviseurs ( $1 * 3 * 9 = 27$ ) n'est pas égal à une puissance de 9, en effet  $9^1 < 27 < 9^2$

- ④ Le nombre **10** est **multipotent** car :

Le produit de ses diviseurs ( $1 * 2 * 5 * 10 = 100$ ) est égal à  $10^2 = 100$ .

- ⑤ Le nombre **16** n'est pas **multipotent** car :

Le produit de ses diviseurs ( $1 * 2 * 4 * 8 * 16 = 1024$ ) n'est pas égal à une puissance de 16, en effet  $16^2 < 1024 < 16^3$

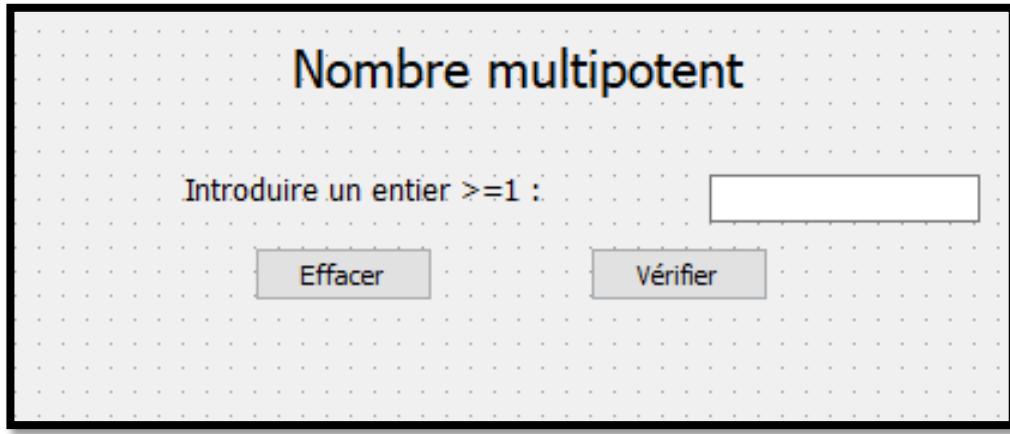
- ⑥ Le nombre **84** est **multipotent** car :

Le produit de ses diviseurs ( $1 * 2 * 3 * 4 * 6 * 7 * 12 * 14 * 21 * 28 * 42 * 84 = 351\ 298\ 031\ 616$ )

est égal à  $84^6 = 351\ 298\ 031\ 616$ .

Pour vérifier si un entier naturel non nul N est un nombre **multipotent** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombre multipotent**"
- Un label demandant la saisie du nombre :  
"Introduire un entier  $\geq 1$  :"
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- 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 "**InterMULT**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**PgmeMULT**".
- 3) Développer dans le programme "**PgmeMULT**", une fonction **Multipotent (N)** qui permet de vérifier si l'entier **N** saisi est **multipotent** ou non.
- 4) Dans le programme "**PgmeMULT**" :
  - ◎ Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterMULT**" 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 **Multipotent** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterMULT**".

**NB :** L'affichage du message doit être conforme aux exemples d'exécution suivants :

## Exemples d'exécution

Nombre multipotent

Introduire un entier  $\geq 1$  :

Désolé ! vérifier la saisie du nombre !!

Nombre multipotent

Introduire un entier  $\geq 1$  :

Désolé ! vérifier la saisie du nombre !!

Nombre multipotent

Introduire un entier  $\geq 1$  :

Le nombre 16 n'est pas multipotent

Nombre multipotent

Introduire un entier  $\geq 1$  :

75 est un nombre multipotent

Nombre multipotent

Introduire un entier  $\geq 1$  :

Le nombre 9 n'est pas multipotent

Nombre multipotent

Introduire un entier  $\geq 1$  :

48 est un nombre multipotent

## 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_()
```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre lisse

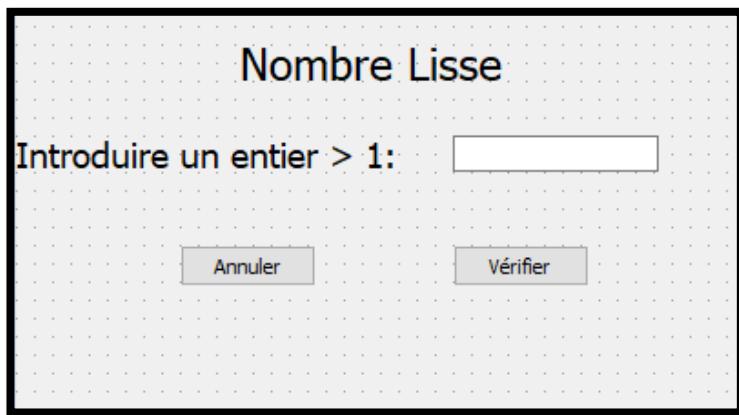
Un nombre **N** est dit **lisse** lorsque son **plus grand diviseur premier** est inférieur ou égal à la racine carrée du nombre **N**.

**Exemples :**

- **N = 60**, les diviseurs de **60** sont : **1, 2, 3, 4, 5, 6, 10, 12, 15, 30** et **60**. Son plus grand diviseur premier est **5** et puisque  $5 \leq \sqrt{60}=7.746\dots$ , donc **60 est un nombre lisse**.
- **N = 49**, les diviseurs de **49** sont : **1, 7 et 49**. Son plus grand diviseur premier est **7** et puisque  $7 \leq \sqrt{49}=7$  donc **49 est un nombre lisse**.
- **N = 22**, les diviseurs de **22** sont : **1, 2, 11 et 22**. Son plus grand diviseur premier est **11** et puisque  $11 > \sqrt{22}=4.690\dots$ , donc **22 n'est pas un nombre lisse**.

Pour vérifier si un entier naturel **N** (**N > 1**) est un nombre **lisse** ou non, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "**Nombre lisse**"
- Un label demandant la saisie d'un nombre : " **Introduire un entier > 1 :** "
- Une zone de saisie permettant la saisie du nombre.
- Un bouton intitulé " **Vérifier**"
- Un bouton intitulé " **Effacer**" (permettant d'effacer la zone de saisie et le label d'affichage)
- 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 "**InterfaceLisse**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**NbrLisse**".

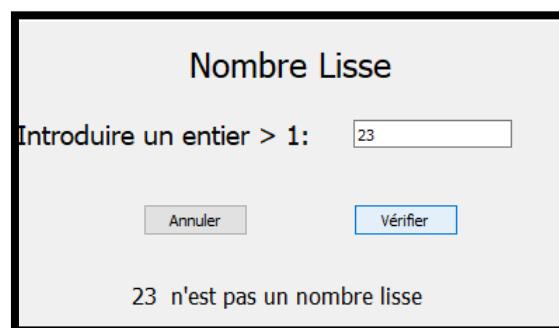
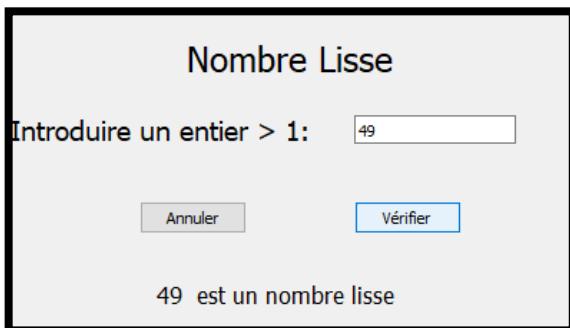
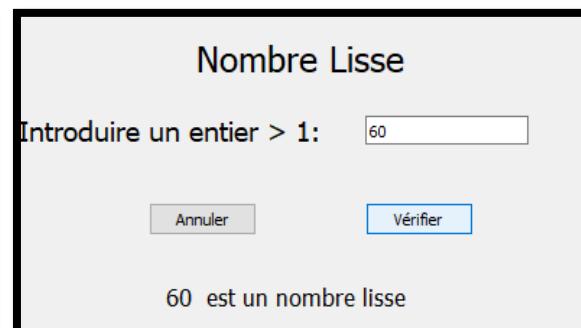
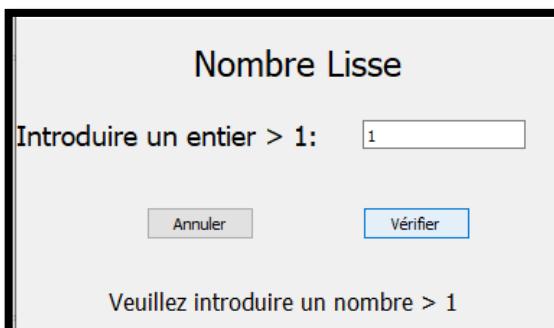
3) Développer, dans le programme "NbrLisse", une fonction **Lisse(N)** qui permet de vérifier si un entier **N** est lisse ou non.

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

- ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterfaceLisse**" 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 "**Lisse**" afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterfaceLisse**".

**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 :



**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_()
```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre semi premier

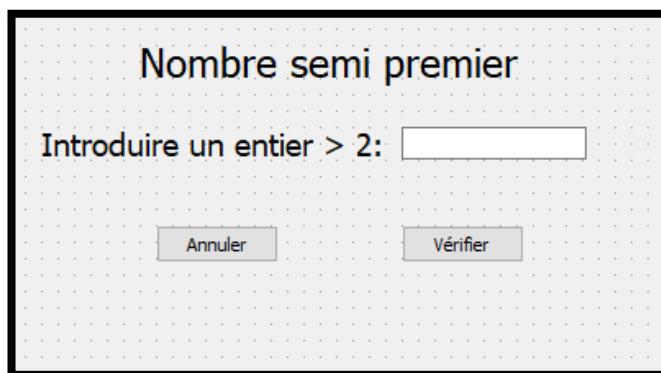
Un nombre N est dit **semi-premier** lorsqu'il est égal au produit de **deux nombres premiers** non nécessairement distincts. C'est-à-dire  $N = k \times k$  avec k est un nombre premier ou  $N = k \times j$  avec k et j sont deux nombres premiers.

**Exemples :**

- **6 est un nombre semi-premier** car  $6 = 2 \times 3$  avec 2 et 3 sont deux nombres premiers.
- **25 est un nombre semi-premier** car  $25 = 5 \times 5$  avec 5 est un nombre premier.
- **831 est un nombre semi-premier** car  $831 = 3 \times 277$  avec 3 et 277 sont deux nombres premiers
- **8 n'est pas un nombre semi-premier**, car  $8 = 2 \times 4$  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 bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- 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 :

Nombre semi premier

Introduire un entier > 2: 2

Veuillez introduire un nombre > 2

Annuler Vérifier

Nombre semi premier

Introduire un entier > 2: 831

831 est semi premier

Annuler Vérifier

Nombre semi premier

Introduire un entier > 2: 16

16 n'est pas semi premier

Annuler Vérifier

Nombre semi premier

Introduire un entier > 2: 15

15 est semi premier

Annuler Vérifier

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_()
```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Le nombre hautement abondant

Un nombre **N** est dit **hautement abondant** si la somme de ses diviseurs (lui-même inclus) est strictement supérieure à la somme des diviseurs de n'importe quel nombre plus petit que lui (le nombre inclus).

### Exemples :

- **8** est hautement abondant car la somme de ses diviseurs, qui est égale à **15** (**1+2+4+8=15**), est strictement supérieure à la somme des diviseurs de tout entier plus petit que lui.
- **5** n'est pas hautement abondant car la somme de ses diviseurs, qui est égale à **6** (**1+5=6**), est inférieure à **7** qui la somme des diviseurs de **4** (**1+2+4=7**).

Soit l'algorithme de la fonction **SOMDIV** suivant qui permet de calculer la somme des diviseurs d'un entier naturel non nul.

**Fonction SOMDIV (n : Entier): Entier**

**DEBUT**

**S ← n**

**Pour i de 1 à n DIV 2 faire**

**Si n MOD i = 0 alors**

**S ← S + i**

**Fin Si**

**Fin Pour**

**Retourner S**

**FIN**

On décide afficher tous les nombres hautement abondants compris entre deux bornes **p** et **q** saisies avec **3<p<q<50** en faisant appel à la fonction **SOMDIV**, on se propose de concevoir une interface graphique contenant les éléments suivants :

- Un label contenant le texte : "Nombres hautement abondants"
- Un label demandant la saisie de la borne inférieure : "Introduire un entier p >3 : "
- Une zone de saisie permettant la saisie de la borne inférieure.
- Un label demandant la saisie de la borne supérieure : "Introduire un entier q tel que p <q<50 : "
- Une zone de saisie permettant la saisie de la borne supérieure.
- Un bouton intitulé "Vérifier"
- Un bouton intitulé "Effacer" (permettant d'effacer la zone de saisie et le label d'affichage)
- Un label pour afficher le message adéquat

**Nombres hautement abondants**

Introduire un entier  $p > 3$ :

Introduire un entier  $q$  tel que  $p < q < 50$ :

## Travail demandé :

- 1) Concevoir une interface graphique comme illustrée ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**InterHA**".
- 2) Créer un programme Python et l'enregistrer, dans votre dossier de travail, sous le nom "**NbrHA**".
- 3) Développer dans le programme "**NbrHA**", une fonction **NbresHA (p, q)** qui permet déterminer les nombres **hautement abondants** comprises entre **p** et **q**.
- 4) Dans le programme "**NbrHA**" :
  - Ⓐ Ajouter les instructions permettant d'appeler l'interface graphique intitulée "**InterHA**" en exploitant **l'annexe** ci-après.
  - Ⓑ Développer un module "**Play**", qui s'exécute suite à un clic sur le bouton "**Vérifier**", permettant de récupérer les deux bornes (**p** et **q**) saisis, puis d'exploiter la fonction **NbresHA** afin d'afficher le message adéquat via le **label** dédié à l'affichage de l'interface "**InterHA**".

**Nombres hautement abondants**

Introduire un entier  $p > 3$ :

Introduire un entier  $q$  tel que  $p < q < 50$ :

Veuillez introduire les deux bornes satisfaisantes les conditions !!

**Nombres hautement abondants**

Introduire un entier  $p > 3$ :

Introduire un entier  $q$  tel que  $p < q < 50$ :

20,24,30,36,42

**Nombres hautement abondants**

Introduire un entier  $p > 3$ :

Introduire un entier  $q$  tel que  $p < q < 50$ :

4,6,8,10,12,16,18,20,24,30

**Nombres hautement abondants**

Introduire un entier  $p > 3$ :

Introduire un entier  $q$  tel que  $p < q < 50$ :

4,6,8,10,12,16,18,20,24,30,36,42,48

**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_()
```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Jeu de chance

Dans le cadre d'une campagne publicitaire, une société commerciale a décidé d'organiser chaque semaine un jeu de chance pour ses clients.

Le principe du jeu consiste à calculer le nombre de chance à partir du numéro de téléphone du client donné et d'afficher le message "**Félicitations, vous avez gagné.**" dans le cas où ce nombre est **premier** ou le message "**Désolé, vous n'avez pas gagné.**" dans le cas contraire.

Sachant que :

- Le numéro de téléphone devrait commencer par 2, 4, 5 ou 9.
- Le nombre de chance est la somme de chaque chiffre du numéro de téléphone multiplié par son indice avec l'indice du premier chiffre est 0.
- Un nombre premier est un nombre qui est divisible par 1 et par lui-même.

**Exemple :**

Donner le numéro : **29234560**

Le programme affiche : **Désolé, vous n'avez pas gagné.**

En effet, le nombre de chance est égal à 99 qui n'est pas un nombre premier.

$$99 = 2 * 0 + 9 * 1 + 2 * 2 + 3 * 3 + 4 * 4 + 5 * 5 + 6 * 6 + 0 * 7$$

c'est la somme de chaque chiffre du

numéro de téléphone multiplié par son indice :

<b>Numéro téléphone</b>	2	9	2	3	4	5	6	0
<b>Indice</b>	0	1	2	3	4	5	6	7

Ci-après, un algorithme de la fonction "**Chance**" à exploiter pour résoudre le problème posé.

### Fonction Chance (Ch : Chaine) : Chaine

**DEBUT**

**Si NON** ( Estnum ( Ch ) **ET** long ( Ch ) = 8 **ET** Ch[0] ∈ ["2", "4", "5", "9"] ) **Alors**

    msg ← "Vérifier le numéro de téléphone !"

**Sinon**

    msg ← "Désolé, vous n'avez pas gagné."

    s ← 0

**Pour i de 0 à long ( Ch ) - 1 Faire**

        s ← s + valeur ( Ch [i] ) \* i

**Fin Pour**

**Si** premier (s) **Alors**

        msg ← "Félicitation, vous avez gagné."

**FinSi**

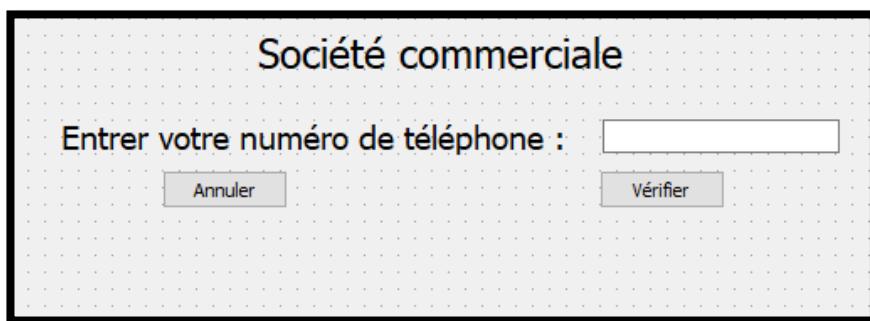
**FinSi**

**Retourner** msg

**FIN**

La société a décidé de créer l'interface graphique présentée ci-dessus, comportant les éléments suivants :

- Un label contenant le nom de la société.
- Un label demandant la saisie du numéro de téléphone.
- Une zone de saisie permettant la saisie du numéro de téléphone.
- Un bouton nommé "**Jouer**".
- Un label pour afficher un message.



### Travail demandé :

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface\_Jeu**".
- 2) Implémenter en Python la fonction "**Chance**" dans un programme et l'enregistrer sous le nom "**Jeu0**", dans votre dossier de travail.
- 3) Développer la fonction "**Premier**" permettant de vérifier si un nombre, passé comme paramètre, est premier ou non puis l'enregistrer dans votre dossier de travail sous le nom "**Jeu1**".
- 4) Dans le programme "**Jeu1**", ajouter les instructions permettant :
  - D'appeler l'interface graphique intitulée "**Interface\_Jeu**" en exploitant l'annexe ci-dessous.

- D'implémenter un module "**Play**", qui s'exécute à la suite d'un clic sur le bouton "**Jouer**", permettant de récupérer le numéro de téléphone saisi puis d'exploiter la fonction "**Chance**" afin d'afficher le message retourné via un **label** de l'interface "**Interface\_Jeu**".

## **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_()
```

**Important :**

1. Une solution modulaire au problème posé est exigée.
2. Dans le répertoire **Bac2023**, créez un dossier de travail ayant comme nom votre numéro d'inscription (**6chiffres**) et dans lequel vous devez enregistrer, au fur et à mesure, tous les fichiers solution à ce sujet.

## Salle de sport

Un gérant d'une salle de sport veut récompenser les adhérents fidèles en leur offrant un bonus sous forme d'heures d'entraînement gratuites, calculé à partir de leurs numéros d'abonnement.

Le bonus est calculé en fonction de l'ancienneté de l'adhérent, exprimée en nombre de mois par rapport à la date du jour. En effet, une heure supplémentaire est offerte pour chaque mois d'ancienneté, sachant que le bonus ne sera pris en considération que si l'ancienneté dépasse 5 ans.

Un numéro d'abonnement est formé de **10** chiffres répartis comme suit :

- Les 4 premiers chiffres représentent l'année d'adhésion qui doit être comprise entre 2000 et 2022.
- Les 2 suivants représentent le mois d'adhésion dont la valeur doit être comprise entre 1 et 12.
- Les 4 derniers chiffres représentent le numéro d'adhésion qu'on suppose distinct pour tous les adhérents.

### Exemple :

Pour le numéro d'abonnement **2016020110**, l'adhérent est donc l'année d'adhésion est **2016**, le mois d'adhésion est **02 (Février)** et son numéro d'adhésion est **0110**. Le bonus accordé à cet adhérent est de **86** heures. En effet, son ancienneté est égale à six ans et deux mois par rapport à la date d'aujourd'hui (**02/05/2023**), en nombre de mois elle est égale à **86** (**12\*7 + 2**).

Notre objectif est de créer un programme qui reçoit en entrée numéro d'abonnement et qui doit afficher le bonus accordé à cet adhérent sachant que la date du jour à considérer est **02/05/2023**.

Ci-après, un algorithme de la fonction "**Fidelite**" à exploiter pour résoudre le problème posé.

#### **Fonction Fidelite (Ch : chaîne): Chaîne**

**DEBUT**

**Si NON Valide (ch)Alors**

msg← "Vérifier le numéro d'abonnement"

**Sinon Si Anciennete (ch)<60 alors**

msg← "L'ancienneté de l'adhérent est inférieur à 5 ans"

**Sinon**

msg← "L'adhérent à un bonus de " + convch(Anciennete (ch)) +" heures"

**Fin Si**

**Retourner msg**

**FIN**

On désire créer l'interface graphique présentée ci-dessous, comportant les éléments suivants :

- Un label contenant le titre "**Salle de sport**".
- Un label demandant la saisie d'un numéro d'adhésion.
- Une zone de saisie permettant la saisie de numéro d'adhésion.
- Un bouton nommé "**Calculer**".
- Un label pour afficher un message.



### Travail demandé :

- 1) Concevoir une interface graphique comme illustré ci-dessus et l'enregistrer, dans votre dossier de travail, sous le nom "**Interface\_adhesion**".
- 2) Implémenter en Python la fonction "**Fidelite**" dans un programme et l'enregistrer sous le nom "**Bonus0**", dans votre dossier de travail.
- 3) Développer la fonction **Valide** qui permet de vérifier si un numéro d'abonnement, passé comme paramètre, est valide ou non puis l'enregistrer dans votre dossier de travail sous le nom "**Bonus1**".
- 4) Dans le programme "**Bonus1**", développer la fonction "**Anciennete**" qui prend comme paramètre un numéro d'abonnement et qui permet de calculer le nombre de mois d'ancienneté de l'adhérent.
- 5) Dans le programme "**Bonus1**", ajouter les instructions permettant:
  - a. D'appeler l'interface graphique intitulée "**Interface\_adhesion**" en exploitant l'annexe ci-dessous.
  - b. D'implémenter un module "**Bonus**", qui s'exécute à la suite d'un clic sur le bouton "**Calculer**", permettant de récupérer le numéro d'abonnement et puis d'exploiter la fonction "**Fidelite**" afin d'afficher le message retourné via un **label** de l'interface "**Interface\_adhesion**".

### 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_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 from random import randint
5 from numpy import array
6 T=array([int()]*8)
7 def REMP_AFFICHE(T):
8     for i in range(7):
9         T[i]=randint(1,99)
10        print(T[i], end=" | ")
11    print()
12 ****
13 def GainPerd(N):
14     s=Somme(N)
15     while len(str(s)) !=1:
16         s=Somme(str(s))
17     return Existence(s*s,T)
18 ****
19 def Somme(x):
20     som=0
21     for i in range(len(x)):
22         som+=int(x[i])
23     return som
24 ****
25 def Existence(nb,T):
26     i=-1
27     while i < 8 and T[i] != nb:
28         i+=1
29     return i < 8
30 ****
31 def Play():
32     N=fen.a.text()
33     if not ( N.isdecimal() and ( len(N) ==8 ) and (N[0] in {"2","3","4","5","9"}) ):
34         fen.res.setText("Désolé ! vérifier le numéro de téléphone du client !!")
35     elif GainPerd(N):
36         fen.res.setText("Le client ayant le numéro de téléphone "+N+" \n a gagné un chariot gratuit")
37     else:
38         fen.res.setText("Le client ayant le numéro de téléphone "+N+" \n doit payer les achats du chariot")
39
40 ****
41 def efface():
42     fen.a.clear()
43     fen.res.clear()
44 ****
45 REMP_AFFICHE(T)
46 app=QApplication([])
47 fen=loadUi("InterCHARIOT.ui")
48 fen.show()
49 fen.b1.clicked.connect(Play)
50 fen.b2.clicked.connect(efface)
51 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 #####
4 from random import randint
5 def Generer():
6     ch=""
7     for i in range(26):
8         n=randint(65,90)
9         while ch.find(chr(n)) != -1 or n == i:
10             n=randint(65,90)
11         ch+=chr(n)
12     return ch
13 #####
14 def Crypter(m,c):
15     ch=""
16     for i in range(len(m)):
17         if "A" <= m[i] <= "Z":
18             ch+=c[ord(m[i])-65]
19         else:
20             ch+=m[i]
21     return ch
22 #####
23 def Majuscule_Espace(ch):
24     i=0
25     while i < len(ch) and ("A" <= ch[i] <= "Z" or ch[i]==" "):
26         i+=1
27     return i==len(ch)
28 #####
29 def Play():
30     m=fen.a.text()
31     if not ( 1 <= len(m) <= 100 and Majuscule_Espace(m) ):
32         fen.res.setText("Désolé ! vérifier le message à crypter !!")
33     else:
34         cle=Generer()
35         MC=Crypter(m,cle)
36         fen.cle.setText("la clé générée est: " + cle)
37         fen.res.setText("Le message crypté sera: " + MC)
38 #####
39 def efface():
40     fen.a.clear()
41     fen.cle.clear()
42     fen.res.clear()
43 #####
44 app=QApplication([])
45 fen=loadUi("InterCRYPTE_AC.ui")
46 fen.show()
47 fen.b1.clicked.connect(Play)
48 fen.b2.clicked.connect(efface)
49 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def Crypter(ch):
5     ch_crypté=""
6     nb=1
7     for i in range(len(ch)-1):
8         if ch[i]==ch[i+1]:
9             nb+=1
10        else:
11            ch_crypté+=str(nb)+ch[i]
12            nb=1
13        ch_crypté+=str(nb)+ch[len(ch)-1]
14    return ch_crypté
15 ****
16 ****
17 def Alphabetique(ch):
18     i=0
19     while i < len(ch) and ("A" <= ch[i].upper() <= "Z"):
20         i+=1
21     return i==len(ch)
22 ****
23 def Play():
24     CH=fen.a.text()
25     if not ( 1 <= len(CH) <= 50 and Alphabetique(CH) ):
26         fen.res.setText("Désolé ! vérifier la chaîne à crypter !!")
27     else:
28         CHC=Crypter(CH)
29         fen.res.setText("La chaîne cryptée sera: " + CHC)
30 ****
31 def efface():
32     fen.a.clear()
33     fen.res.clear()
34 ****
35 app=QApplication([])
36 fen=loadUi("InterCRYPTE_OS.ui")
37 fen.show()
38 fen.b1.clicked.connect(Play)
39 fen.b2.clicked.connect(efface)
40 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def occurrence(c,ch):
5     nb=0
6     for i in range(len(ch)):
7         if ch[i]==c:
8             nb+=1
9     return nb
10 ****
11 def Crypter(ch):
12     m=""
13     for i in range(len(ch)):
14         n=occurrence(ch[i],ch)
15         if n % 2==0:
16             k= n // 2
17         else:
18             k=2*n
19         m+=chr( (ord(ch[i]) - ord("A") + k ) % 26 + ord("A"))
20     return m
21 ****
22 def Majuscule(ch):
23     i=0
24     while i < len(ch) and ("A" <= ch[i] <= "Z"):
25         i+=1
26     return i==len(ch)
27 ****
28 def Play():
29     MS=fen.a.text()
30     if not ( 1 <= len(MS) and Majuscule(MS) ):
31         fen.res.setText("Désolé ! vérifier le mot à crypter !!")
32     else:
33         fen.res.setText("Le mot crypté sera: " + Crypter(MS))
34 ****
35 def efface():
36     fen.a.clear()
37     fen.res.clear()
38 ****
39 app=QApplication([])
40 fen=loadUi("InterCRYPTE_AM.ui")
41 fen.show()
42 fen.b1.clicked.connect(Play)
43 fen.b2.clicked.connect(efface)
44 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 #*****
4 def Freres(A,B):
5     voyelle="aeiouy"
6     i=0
7     while i < len(A) and ( (voyelle.find(A[i])==-1 and voyelle.find(B[i])==-1) or
(voyelle.find(A[i]) !=-1 and voyelle.find(B[i])!= -1) ):
8         i+=1
9     return i==len(A)
10 #*****
11 def Cousins(A,B):
12     voyelle="aeiouy"
13     i=0
14     while i < len(A) and ( (voyelle.find(A[i])==-1 and voyelle.find(B[i]) !=-1) or
(voyelle.find(A[i]) !=-1 and voyelle.find(B[i]) ==-1) ):
15         i+=1
16     return i==len(A)
17 #*****
18 def AlphaMinus(ch):
19     i=0
20     while i < len(ch) and "a" <= ch[i] <= "z":
21         i+=1
22     return i==len(ch)
23 #*****
24 def Play():
25     MotA=fen.cha.text()
26     MotB=fen.chb.text()
27     if (MotA=="") or ( not AlphaMinus(MotA) ) or ( len(MotB) != len(MotA) ) or ( not
AlphaMinus(MotB) ):
28         fen.res.setText("Désolé ! vérifier la saisie des mots !!")
29     elif Freres(MotA, MotB):
30         fen.res.setText(MotA + " est frère de "+MotB)
31     elif Cousins (MotA, MotB):
32         fen.res.setText("Le mot "+MotA+" est cousin de "+MotB)
33     else:
34         fen.res.setText("Les deux mots ni frères ni cousins")
35 #*****
36 def efface():
37     fen.cha.clear()
38     fen.chb.clear()
39     fen.res.clear()
40 #*****
41 app=QApplication([])
42 fen=loadUi("InterFRC0.ui")
43 fen.show()
44 fen.b1.clicked.connect(Play)
45 fen.b2.clicked.connect(efface)
46 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 #*****
4 def palindrome(ch):
5     i=0
6     while i<len(ch)//2 and ch[i].upper()==ch[len(ch)-1-i].upper():
7         i+=1
8     return i==len(ch)//2
9 #*****
10 def Est_Voyelles(a,b):
11     voyelle="AEIOUY"
12     if voyelle.find(a.upper()) !=-1 and voyelle.find(b.upper()) !=-1:
13         return True
14     else:
15         return False
16 #*****
17 def Est_Consonnes(a,b):
18     voyelle="AEIOUY"
19     if "A" <= a.upper() <= "Z" and "A" <= b.upper() <= "Z" :
20         if voyelle.find(a.upper()) ==-1 and voyelle.find(b.upper()) ==-1:
21             return True
22         else:
23             return False
24 #*****
25 def Est_Chiffres(a,b):
26     chiffre="0123456789"
27     if chiffre.find(a) !=-1 and chiffre.find(b) !=-1:
28         return True
29     else:
30         return False
31 #*****
32 def Est_Symboles(a,b):
33     if not "A" <= a.upper() <= "Z" and not "0" <= a <= "9" and not "A" <= b.upper() <= "Z" and not
34     "0" <= b <= "9":
35         return True
36     else:
37         return False
38 #*****
39 def Est_PP(ch):
40     verife=not palindrome(ch)
41     i=0
42     while i< len(ch)//2 and verife:
43         if Est_Voyelles(ch[i],ch[len(ch)-1-i]) or Est_Consonnes(ch[i],ch[len(ch)-1-i]) or
44         Est_Chiffres(ch[i],ch[len(ch)-1-i]) or Est_Symboles(ch[i],ch[len(ch)-1-i]):
45             i+=1
46         else:
47             verife=False
48     return verife
49 #*****
50 def Play():
51     ch=fen.a.text()
52     if not (len(ch)!=0 and len(ch) % 2 ==0):
53         fen.res.setText("Désolé ! vérifier la saisie de 1 chaîne !!")
54     elif Est_PP(ch):
55         fen.res.setText("La chaîne "+ch+" est presque palindrome")
56     else:
57         fen.res.setText(ch+" une chaîne n'est pas presque palindrome")
58 #*****
59 def efface():
60     fen.a.clear()
61     fen.res.clear()
```

```
62 | app=QApplication([])
63 | fen=loadUi("InterPP.ui")
64 | fen.show()
65 | fen.b1.clicked.connect(Play)
66 | fen.b2.clicked.connect(efface)
67 | app.exec_()
```

```
1 | from PyQt5.uic import loadUi
2 | from PyQt5.QtWidgets import QApplication
3 | #*****
4 | def Valide(ch):
5 |     #Récupération de deux plus grands facteurs différents telsque p>=2*q
6 |     ch=ch[1:]                                #éliminer le premier "+"
7 |     q=int(ch[:ch.find("+")])                #récupérer le premier facteur
8 |     ch=ch[ch.find["+"]+1:]                  #ch devient sans le premier facteur
9 |     p=int(ch[:ch.find("+")])                #récupérer le second facteur
10 |    ch=ch[ch.find["+"]+1:]                 #ch devient sans le premier et le second facteur
11 |    while ch.find["+"] !=-1:                # tantqu'il ya encore "+"
12 |        q=p                                #l'avant plus grand facteur (q) prend la valeur de
dernier facteur (p)
13 |        p=int(ch[:ch.find("+")])            # le dernier facteur (p) prend le nouveau facteur
14 |        ch=ch[ch.find["+"]+1:]              #la chaîne sans ce dernier facteur
15 |    return p >= 2*q
16 | #*****
17 | def Pointu(N):
18 |     ch="+"
19 |     i=2
20 |     nb=0
21 |     while N !=1:
22 |         if N % i ==0 :
23 |             if ch.find(""+str(i)+"") ==-1:
24 |                 ch+=str(i)+"+"      #concaténer dans ch les facteurs distincts
25 |                 nb+=1
26 |             N/=i
27 |         else:
28 |             i+=1
29 |     return nb==1 or Valide(ch)
30 | #*****
31 | def Play():
32 |     ch=fen.a.text()
33 |     if not ( ch.isdecimal() and ( int(ch) >1 ) ):
34 |         fen.res.setText("Désolé ! vérifier la saisie du nombre !!")
35 |     elif Pointu(int(ch)):
36 |         fen.res.setText(ch + " est un nombre pointu")
37 |     else:
38 |         fen.res.setText("Le nombre "+ ch +" n'est pas pointu")
39 |
40 | #*****
41 | def efface():
42 |     fen.a.clear()
43 |     fen.res.clear()
44 | #*****
45 | app=QApplication([])
46 | fen=loadUi("InterPointu.ui")
47 | fen.show()
48 | fen.b1.clicked.connect(Play)
49 | fen.b2.clicked.connect(efface)
50 | app.exec_()
```

```

1 | from PyQt5.uic import loadUi
2 | from PyQt5.QtWidgets import QApplication
3 | ****
4 | def Valide(ch):
5 |     #Récupération de deux plus grands facteurs différents à partir de la fin de ch
6 |     ch=ch[:len(ch)-1] #éliminer l'étoile de la fin
7 |     nb=0 # désigne le nombre des étoiles à partir de la fin
8 |     i=len(ch)-1 # initialisation du compteur au dernier caractère de ch
9 |     ch1="" # initialisation de ch1 qui va contenir les deux plus grands facteurs
différents séparés
10 |    while not (i== -1 or nb==2): # On arrête le parcours lorsqu'on atteint l'avant début de ch ou
bien le deuxième "*"
11 |        if ch[i]=="+":
12 |            nb+=1 #on incrémente le nb lorsqu'on trouve "*"
13 |            ch1=ch[i]+ch1 #concaténer dans ch1 à l'envers du parcours pourqu'on puisse trouver les
2 plus grands facteurs distincts
14 |            i-=1 # décrémenter le compteur i
15 |            ch1=ch1[1:] #éliminer le premier "*" de ch1 pourqu'elle contient p et q séparés par
un seul "*"
16 |    return (i== -1 and nb<2) or int(ch1[:ch1.find("*")])*2 <= int(ch1[ch1.find("*")+1:])
17 | ****
18 | def Pointu(N):
19 |     ch =""
20 |     i=2
21 |     while N !=1:
22 |         if N % i ==0 :
23 |             if ch.find("*"+str(i)+"*") ==-1:
24 |                 ch+=str(i)+"*" #concaténer dans ch les facteurs distincts
25 |                 N//=i
26 |             else:
27 |                 i+=1
28 |     return Valide(ch)
29 | ****
30 | def Play():
31 |     ch=fen.a.text()
32 |     if not ( ch.isdecimal() and ( int(ch) >1 ) ):
33 |         fen.res.setText("Désolé ! vérifier la saisie du nombre !!")
34 |     elif Pointu(int(ch)):
35 |         fen.res.setText(ch +" est un nombre pointu")
36 |     else:
37 |         fen.res.setText("Le nombre "+ ch +" n'est pas pointu")
38 |
39 | ****
40 | def efface():
41 |     fen.a.clear()
42 |     fen.res.clear()
43 | ****
44 | app=QApplication([])
45 | fen=loadUi("InterPointu.ui")
46 | fen.show()
47 | fen.b1.clicked.connect(Play)
48 | fen.b2.clicked.connect(efface)
49 | app.exec_()

```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def Special(N):
5     s= (N //1000) * (N//10%10) + (N //100%10) * (N%10)
6     i=1
7     while i*i< s:
8         i+=1
9     return i*i==s
10 ****
11 def Play():
12     N=fen.a.text()
13     if not ( N.isdecimal() and ( 1111 <= int(N) <= 9999) and N.find("0") ==-1 ) :
14         fen.res.setText("Désolé ! vérifier la saisie du nombre !!")
15     elif Special(int(N)):
16         fen.res.setText(N +" est un nombre spécial")
17     else:
18         fen.res.setText("Le nombre "+ N +" n'est pas spécial")
19 ****
20 ****
21 def efface():
22     fen.a.clear()
23     fen.res.clear()
24 ****
25 app=QApplication([])
26 fen=loadUi("InterSPEC.ui")
27 fen.show()
28 fen.b1.clicked.connect(Play)
29 fen.b2.clicked.connect(efface)
30 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def Tot_impair(N):
5     p=1
6     i=len(N)-1
7     verife=True
8     while i>-1 and verife:
9         if int(N[i]) % 2 == 1 and p % 2 ==0:
10             verife=False
11         p+=1
12         i-=1
13     return verife
14 ****
15 def Play():
16     N=fen.a.text()
17     if not ( N.isdecimal() and ( int(N) >0) ):
18         fen.res.setText("Désolé ! vérifier la saisie du nombre !!")
19     elif Tot_impair(N):
20         fen.res.setText(N +" est un nombre totalement impair")
21     else:
22         fen.res.setText("Le nombre "+ N +" n'est pas totalement impair")
23 ****
24 def efface():
25     fen.a.clear()
26     fen.res.clear()
27 ****
28 app=QApplication([])
29 fen=loadUi("InterTOTIMP.ui")
30 fen.show()
31 fen.b1.clicked.connect(Play)
32 fen.b2.clicked.connect(efface)
33 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4
5 def Palindrome(N):
6     i = 0
7     while i < len(N) // 2 and N[i] == N[len(N)-1-i] :
8         i = i + 1
9     return i==len(N)// 2
10
11 def Binaire(N):
12     ch=""
13     N=int(N)
14     while N !=0:
15         ch=str(N %2)+ch
16         N=N // 2
17     return ch
18
19
20 def play():
21     N=fen.a.text()
22     if not N.isdecimal() or int(N)==0 or not Palindrome(N):
23         fen.c.setText("Désolé! Introduire un nombre valide !!!")
24     elif Palindrome(Binaire(N)):
25         fen.c.setText("Le nombre "+N+" est doublement palindrome")
26     else:
27         fen.c.setText(N+" n'est pas un nombre doublement palindrome")
28
29
30 def efface():
31     fen.a.clear()
32     fen.c.setText("")
33
34 ****
35 app=QApplication([])
36 fen=loadUi("InterDP.ui")
37 fen.show()
38 fen.b.clicked.connect(play)
39 fen.b2.clicked.connect(efface)
40 app.exec_()
41
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def SommeChiffres(n):
5     ch=str(n)
6     SOM=0
7     for i in range(len(ch)):
8         som+=int(ch[i])
9     return som
10 ****
11 def Smith(N):
12     m=SommeChiffres(N)
13     i=2
14     s=0
15     while N !=1:
16         if N % i ==0:
17             s+=SommeChiffres(i)
18             N//=i
19         else:
20             i+=1
21     return s==m
22 ****
23 def Play():
24     ch=fen.a.text()
25     if not ( ch.isdecimal() and ( int(ch) >1 ) ):
26         fen.res.setText("Désolé ! vérifier la saisie du nombre !!")
27     elif Smith(int(ch)):
28         fen.res.setText(ch +" est un nombre de Smith")
29     else:
30         fen.res.setText("Le nombre "+ ch +" n'est pas Smith")
31
32 ****
33 def efface():
34     fen.a.clear()
35     fen.res.clear()
36 ****
37 app=QApplication([])
38 fen=loadUi("InterSmith.ui")
39 fen.show()
40 fen.b1.clicked.connect(Play)
41 fen.b2.clicked.connect(efface)
42 app.exec_()
```

```
1 from numpy import array
2 ****
3 def saisie():
4     global m
5     m=0
6     while not (4 <= m <= 20):
7         m=int(input("m="))
8 ****
9 def remplir(t,m):
10    for i in range(m):
11        while not (t[i]>=3):
12            t[i]=int(input("t["+str(i)+"]="))
13 ****
14 def afficher(t,m):
15     ch ","
16     for i in range(m):
17         if sectionnable(t[i]) and (ch.find(", "+str(t[i])+",") == -1):
18             ch+=str(t[i])+","
19     if ch != ",":
20         print("Les nombres sectionnable unitaires sont:",ch[1:len(ch)-1])
21     else:
22         print("Aucun nombre sectionnable unitaire dans le tableau")
23 ****
24 def sectionnable(x):
25     s=1
26     i=1
27     while not ( s >= x ):
28         i+=1
29         s+=i
30     return s==x
31 ****
32
33 #-----PP-----
34 ****
35 saisie()
36 t=array([int()]*m)
37 remplir(t,m)
38 afficher(t,m)
39
40
```

```
1 from numpy import array
2 ****
3 def saisie():
4     global m
5     m=0
6     while not (4 <= m <= 20):
7         m=int(input("m="))
8 ****
9 def remplir(t,m):
10    for i in range(m):
11        while not (t[i]>=10):
12            t[i]=int(input("t["+str(i)+"]="))
13 ****
14 def afficher(t,m):
15    ch=""
16    for i in range(m):
17        if etrange(t[i]):
18            ch+=str(t[i])+","
19    if ch != "":
20        print("Les nombres ETRANGES sont:",ch[:len(ch)-1])
21    else:
22        print("Aucun nombre ETRANGE dans le tableau")
23 ****
24 def etrange(x):
25    ch=str(x)
26    i=1
27    while not ( ( i==len(ch) ) or ( int(ch[:i+1]) % len(ch[:i+1]) !=0 ) ):
28        i+=1
29    return i == len(ch)
30 ****
31
32 #-----PP-----
33 ****
34 saisie()
35 t=array([int()]*m)
36 remplir(t,m)
37 afficher(t,m)
38
```

```
1 from numpy import array
2 #*****
3 def saisie():
4     global N
5     N=0
6     while not (3 <= N <= 20):
7         N=int(input("N="))
8 #*****
9 def remplir(TN,TT,N):
10    for i in range(N):
11
12        TN[i]=input("TN["+str(i)+"]=")
13        while TN[i] == "" or not Alphabetique_Espace(TN[i]) or not Distinct(TN[i],TN,i) :
14            TN[i]=input("TN["+str(i)+"]=")
15
16        TT[i]=int(input("TT["+str(i)+"]="))
17        while TT[i] <= 0 :
18            TT[i]=int(input("TT["+str(i)+"]="))
19 #*****
20 def Alphabetique_Espace(ch):
21     #on vérifie que l'espace n'existe ni au début ni à la fin de ch ainsi que l'inexistence de
double espace aussi
22     verife= ("A" <= ch[0].upper() <= "Z") and ("A" <= ch[len(ch)-1].upper() <= "Z") and (ch.find(" ") == -1)
23     i=1
24     while i<len(ch)-1 and verife:
25         if ("A" <= ch[i].upper() <= "Z") or ch[i]==" ":
26             i+=1
27         else:
28             verife=False
29     return verife
30 #*****
31 def Distinct(ch, TN,p):
32     i=0
33     while TN[i] != ch:
34         i+=1
35     return i==p
36 #*****
37 def afficher(TN,TT,N):
38     Trier(TN,TT,N)
39     print(TN)
40     print(TT)
41     print("Le classement est:")
42     print("Rang n°1:")
43     print("\t L'équipe",TN[0]," a pris ", Temps(TT[0])," pour réaliser toutes les épreuves")
44
45     for i in range(1,N):
46         if TT[i] != TT[i-1]:
47             print("Rang n°",i+1,":")
48             print("\t L'équipe",TN[i]," a pris ", Temps(TT[i])," pour réaliser toutes les épreuves")
49
50 #*****
51 def Trier(TN,TT,N):
52     for i in range(N-1):
53         IndMin=i
54         for j in range(i+1,N):
55             if TT[j] < TT[IndMin]:
56                 IndMin=j
57         if IndMin != i:
58             TN[i],TN[IndMin]=TN[IndMin],TN[i]
59             TT[i],TT[IndMin]=TT[IndMin],TT[i]
60
61 #*****
```

```
62 | def Temps(M):
63 |     return str(M // 3600) + " H :" + str(M % 3600 // 60) + " Min :" + str(M % 60) + " Sec"
64 | ****
65 |
66 |-----PP-----
67 | ****
68 | saisie()
69 | TN=array([str]*N)
70 | TT=array([int()]*N)
71 | remplir(TN, TT, N)
72 | afficher(TN, TT, N)
73 |
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def CRODEC(n):
5     i=len(n)//2 -1
6     while i>= 0 and n[i] < n[i+1] and n[len(n)-i-1] < n[len(n)-i-2]:
7         i-=1
8     return i== -1
9 -----
10 def DECRO(n):
11     i=len(n)//2 -1
12     while i>= 0 and n[i] > n[i+1] and n[len(n)-i-1] > n[len(n)-i-2]:
13         i-=1
14     return i== -1
15 -----
16 def play():
17     n=window.a.text()
18     if not n.isdecimal() or len(n) % 2 ==0 or int(n) <100 :
19         window.c.setText("Veuillez introduire un nombre >= 100 dont le nombre de ses chiffres est
impair")
20     elif CRODEC(n):
21         window.c.setText(n+" est un nombre CRO-DEC")
22     elif DECRO(n):
23         window.c.setText(n+" est un nombre DEC-CRO")
24     else:
25         window.c.setText(n+" ni CRO-DEC ni DEC-CRO")
26 ****
27 app=QApplication([])
28 window=loadUi("InterCRODEC.ui")
29 window.show()
30 window.b.clicked.connect(play)
31 app.exec_()
32
33
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def Emprisonnierz(A,B):
5     i=0
6     while i< len(A) and B.find(A[i])!=-1:
7         B=B[B.find(A[i])+1:]
8         i+=1
9     return i==len(A)
10 ****
11 def AlphaMajus(ch):
12     i=0
13     while i < len(ch) and "A" <= ch[i] <= "Z":
14         i+=1
15     return i==len(ch)
16 ****
17 def Play():
18     MotA=fen.cha.text()
19     MotB=fen.chb.text()
20     if (MotA=="") or ( not AlphaMajus(MotA) ) or ( len(MotB) < len(MotA) ) or ( not
AlphaMajus(MotB) ):
21         fen.res.setText("Désolé ! vérifier la saisie des mots !!")
22     elif Emprisonnierz(MotA, MotB):
23         fen.res.setText(MotA + " est prisonnié dans le mot "+MotB)
24     else:
25         fen.res.setText("Le mot "+MotA+" n'est pas prisonnié dans le mot "+MotB)
26 ****
27 def efface():
28     fen.cha.clear()
29     fen.chb.clear()
30     fen.res.clear()
31 ****
32 app=QApplication([])
33 fen=loadUi("InterEMPRIS.ui")
34 fen.show()
35 fen.b1.clicked.connect(Play)
36 fen.b2.clicked.connect(efface)
37 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 #*****
4 def Distinct(ch):
5     x=""
6     while ch.find("*") !=-1:
7         if x.find(ch[:ch.find("*")]) ==-1:
8             x+=ch[:ch.find("*")]+ "*"
9             ch=ch[ch.find("*")+1:]
10    return x
11 #*****
12 def FacteursPremiers(N):
13     ch=""
14     i=2
15     while N != 1:
16         if N % i == 0:
17             ch+=str(i)+"*"
18             N=N//i
19         else:
20             i+=1
21     return ch
22 #*****
23 def Homogene(A,B):
24     return Distinct(FacteursPremiers(A)) == Distinct(FacteursPremiers(B))
25 #*****
26 def Play():
27     cha=fen.a.text()
28     chb=fen.b.text()
29     if not ( cha.isdecimal() and chb.isdecimal() and ( int(cha) > 1 ) and ( int(chb) > 1 ) and
30             ( int(cha) != int(chb) ) ):
31         fen.res.setText("Désolé ! vérifier la saisie des nombres !!")
32     elif Homogene(int(cha), int(chb)):
33         fen.res.setText(cha + " et "+chb+" sont homogènes")
34     else:
35         fen.res.setText("Les deux nombres "+ cha +" et "+chb+" ne sont pas homogènes")
36 #*****
37 def efface():
38     fen.a.clear()
39     fen.b.clear()
40     fen.res.clear()
41 #*****
42 app=QApplication([])
43 fen=loadUi("InterHOMO.ui")
44 fen.show()
45 fen.b1.clicked.connect(Play)
46 fen.b2.clicked.connect(efface)
47 app.exec_()
48
```

```

1 | from PyQt5.uic import loadUi
2 | from PyQt5.QtWidgets import QApplication
3 | ****
4 | def FacteursDistincts(N):
5 |     chD="" # chaine qui va contenir les facteurs distincts
6 |     i=2
7 |     while N != 1:
8 |         if N % i == 0: # si i est un facteur premier
9 |             N=N//i
10 |             if chD.find("*"+str(i)+"*") ==-1: # si i n'existe pas dans la chaine qui contient les
facteurs distincts
11 |                 chD+=str(i)+"*" # on lui ajoute à chD
12 |             else:
13 |                 i+=1
14 |     return chD
15 | ****
16 | def Homogene(A,B):
17 |     return FacteursDistincts(A)==FacteursDistincts(B)
18 | ****
19 | def Play():
20 |     cha=fen.a.text()
21 |     chb=fen.b.text()
22 |     if not ( cha.isdecimal() and chb.isdecimal() and ( int(cha) > 1 ) and ( int(chb) > 1 ) and
( int(cha) != int(chb) ) ):
23 |         fen.res.setText("Désolé ! vérifier la saisie des nombres !!")
24 |     elif Homogene(int(cha), int(chb)):
25 |         fen.res.setText(cha +" et "+chb+" sont homogènes")
26 |     else:
27 |         fen.res.setText("Les deux nombres "+ cha +" et "+chb+" ne sont pas homogènes")
28 |
29 | ****
30 | def efface():
31 |     fen.a.clear()
32 |     fen.b.clear()
33 |     fen.res.clear()
34 | ****
35 | app=QApplication([])
36 | fen=loadUi("InterHOMO.ui")
37 | fen.show()
38 | fen.b1.clicked.connect(Play)
39 | fen.b2.clicked.connect(efface)
40 | app.exec_()
41 |

```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def est_binaire(n):
5     ch=str(n)
6     k=0
7     while k < len(ch) and ch[k] in {"0","1"}:
8         k+=1
9     return k==len(ch)
10 ****
11 def ppmb(n):
12     j=1
13     while not est_binaire(j*n):
14         j+=1
15     return j*n
16 ****
17 def Play():
18     ch=fen.a.text()
19     if not (ch.isdecimal() and (1 <= int(ch) <= 88) and (int(ch)%9!=0) and ch.find("9")  
==1):
20         fen.res.setText("Désolé ! vérifier la saisie du nombre !!")
21     else:
22         fen.res.setText(ch + " * "+str(ppmb(int(ch))// int(ch))+ " = "+str(ppmb(int(ch))))
23 ****
24 def efface():
25     fen.a.clear()
26     fen.res.clear()
27 ****
28 app=QApplication([])
29 fen=loadUi("InterPPMB.ui")
30 fen.show()
31 fen.b1.clicked.connect(Play)
32 fen.b2.clicked.connect(efface)
33 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def ProduitDiv(N):
5     prod=1
6     for i in range(2,N+1):
7         if N % i ==0:
8             prod*=i
9     return prod
10 ****
11 def Puissance(a,b):
12     p=1
13     for i in range(b):
14         p*=a
15     return p
16 ****
17 def Multipotent(N):
18     p=ProduitDiv(N)
19     i=1
20     while Puissance(N,i) < p:
21         i+=1
22     return Puissance(N,i)==p
23 ****
24 def Play():
25     ch=fen.a.text()
26     if not ( ch.isdecimal() and ( int(ch) >=1 ) ):
27         fen.res.setText("Désolé ! vérifier la saisie du nombre !!")
28     elif Multipotent(int(ch)):
29         fen.res.setText(ch +" est un nombre multipotent")
30     else:
31         fen.res.setText("Le nombre "+ ch +" n'est pas multipotent")
32
33 ****
34 def efface():
35     fen.a.clear()
36     fen.res.clear()
37 ****
38 app=QApplication([])
39 fen=loadUi("InterMULT.ui")
40 fen.show()
41 fen.b1.clicked.connect(Play)
42 fen.b2.clicked.connect(efface)
43 app.exec_()
```

```
1 from PyQt5.QtWidgets import QApplication
2 from PyQt5.QtWidgets import QMainWindow
3 #*****
4 from math import sqrt
5 def Lisse(N):
6     i=N//2
7     verife=False
8     while not (verife or i < 1):
9         if N % i ==0 and premier(i):
10             verife=True
11         else:
12             i-=1
13     return verife and i <=int(sqrt(N))
14 #*****
15 def premier (m):
16     nb=0
17     for i in range(1, m+1):
18         if m % i==0:
19             nb+=1
20     return nb==2
21 #*****
22 def play():
23     N=fen.a.text()
24     if not N.isdecimal() or int(N) <= 1 :
25         fen.c.setText("Veuillez introduire un nombre > 1")
26     elif Lisse(int(N)):
27         fen.c.setText(N+" est un nombre lisse")
28     else:
29         fen.c.setText(N+" n'est pas un nombre lisse")
30 #*****
31 app=QApplication([])
32 fen=loadUi("InterfaceLisse.ui")
33 fen.show()
34 fen.b.clicked.connect(play)
35 app.exec_()
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 from math import sqrt
5 def Lisse(N):
6     i=N//2
7     while not (i==1 or (N % i ==0 and premier(i))):
8         i-=1
9     return 1 < i <=int(sqrt(N))
10 ****
11 def premier (m):
12     if m<=1: return False
13     d=2
14     while d*d <=m:
15         if m % d ==0:
16             return False
17         else:
18             d+=1
19     return True
20 ****
21 def play():
22     N=fen.a.text()
23     if not N.isdecimal() or int(N) <= 1 :
24         fen.c.setText("Veuillez introduire un nombre > 1")
25     elif Lisse(int(N)):
26         fen.c.setText(N+" est un nombre lisse")
27     else:
28         fen.c.setText(N+" n'est pas un nombre lisse")
29 ****
30 app=QApplication([])
31 fen=loadUi("InterfaceLisse.ui")
32 fen.show()
33 fen.b.clicked.connect(play)
34 app.exec_()
35
36
37
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def SemiPremier(N):
5     i=2
6     verife=False
7     while not (verife or i*i > N):
8         if N % i ==0 and premier(i) and premier(N // i):
9             verife=True
10        else:
11            i+=1
12    return verife
13 ****
14 def premier (m):
15     nb=0
16     for i in range(1, m+1):
17         if m % i==0:
18             nb+=1
19     return nb==2
20 ****
21 ****
22 def play():
23     N=fen.nombre.text()
24     if not N.isdecimal() or int(N) <= 2 :
25         fen.lab_res.setText("Veuillez introduire un nombre > 2")
26     elif SemiPremier(int(N)):
27         fen.lab_res.setText(N+" est semi premier")
28     else:
29         fen.lab_res.setText(N+" n'est pas semi premier")
30 ****
31 ****
32 app=QApplication([])
33 fen=loadUi("InterfaceSemiPremier.ui")
34 fen.show()
35 fen.Btverife.clicked.connect(play)
36 app.exec_()
37 
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 ****
4 def SemiPremier(N):
5     i=2
6     nb=0
7     while N !=1 and nb <=2:
8         if N % i ==0 :
9             nb+=1
10            N//=i
11        else:
12            i+=1
13    return nb==2
14
15 ****
16 def play():
17     N=fen.nombre.text()
18     if not N.isdecimal() or int(N) <= 2 :
19         fen.lab_res.setText("Veuillez introduire un nombre > 2")
20     elif SemiPremier(int(N)):
21         fen.lab_res.setText(N+" est semi premier")
22     else:
23         fen.lab_res.setText(N+" n'est pas semi premier")
24
25 ****
26 app=QApplication([])
27 fen=loadUi("InterfaceSemiPremier.ui")
28 fen.show()
29 fen.Btverife.clicked.connect(play)
30 app.exec_()
31
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication
3 #*****
4 def HautementAbondant(N):
5     verife=True
6     s=SOMDIV(N)
7     for i in range(1,N):
8         if s<= SOMDIV(i):
9             verife=False
10    return verife
11 #*****
12 def SOMDIV (m):
13     s=m
14     for i in range(1, (m //2)+1):
15         if m % i==0:
16             s+=i
17     return s
18 #*****
19 def NbresHA(p,q):
20     msg=""
21     for i in range(int(p), int(q)+1):
22         if HautementAbondant(i):
23             msg+=str(i)+","
24     return msg[:len(msg)-1]
25 #*****
26 def play():
27     p=fen.p.text()
28     q=fen.q.text()
29     if not ( p.isdecimal() and int(p)>3 and q.isdecimal() and int(p) < int(q) < 50 ) :
30         fen.c.setText("Veuillez introduire les deux bornes satisfaisantes les conditions !!")
31     else :
32         fen.c.setText(NbresHA(int(p),int(q)))
33 #*****
34 app=QApplication([])
35 fen=loadUi("InterHA.ui")
36 fen.show()
37 fen.b.clicked.connect(play)
38 app.exec_()
39
```

```
1 def Chance (Ch) :
2     if not (Ch.isdecimal() and len ( Ch ) == 8 and Ch[0] in ["2","4","5","9"] ) :
3         msg = "Vérifier le numéro de téléphone !"
4     else :
5         msg = "Désolé, vous n'avez pas gagné."
6         s = 0
7         for i in range (len ( Ch )) :
8             s = s + int ( Ch [i] ) * i
9         if Premier (s) :
10            msg = "Félicitation, vous avez gagné."
11
12 return msg
13 #-----
14
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication,QMessageBox,QTableWidget,QTableWidgetItem
3 #-----
4 def Premier (m):
5     nb=0
6     for i in range(1, m+1):
7         if m % i==0:
8             nb+=1
9     return nb==2
10 #-----
11 def Chance (Ch) :
12     if not (Ch.isdecimal() and len ( Ch ) == 8 and Ch[0] in ["2","4","5","9"] ) :
13         msg = "Vérifier le numéro de téléphone !"
14     else :
15         msg = "Désolé, vous n'avez pas gagné."
16         s = 0
17         for i in range (len ( Ch )) :
18             s = s + int ( Ch [i] ) * i
19         if Premier (s) :
20            msg = "Félicitation, vous avez gagné."
21
22 return msg
23 #-----
24 def Play() :
25     form.res.setText(Chance(form.a.text()))
26 #-----
27 app = QApplication([])
28 form = loadUi ("Interface_Jeu.ui")
29 form.show()
30 form.b.clicked.connect (Play)
31 app.exec_()
32
```

```
1 def Fidelite(ch):
2     if not Valide(ch):
3         msg= "Verifier le numéro d'abonnement"
4     elif Anciennete (ch) < 60:
5         msg ="L'ancienneté de l'adhérent est inférieur à 5 ans"
6     else:
7         msg = "L'adhérent à un bonus de " + str(Anciennete (ch)) +" heures"
8
9     return msg
10
11 #-----
12
```

```
1 from PyQt5.uic import loadUi
2 from PyQt5.QtWidgets import QApplication,QMessageBox,QTableWidget,QTableWidgetItem
3 #-----
4 def Fidelite(ch):
5     if not Valide(ch):
6         msg= "Verifier le numéro d'abonnement"
7     elif Anciennete (ch) < 60:
8         msg ="L'ancienneté de l'adhérent est inférieur à 5 ans"
9     else:
10        msg = "L'adhérent à un bonus de " + str(Anciennete (ch)) +" heures"
11
12     return msg
13 #-----
14 # Vérifier si un numéro d'abonnement est valide
15 def Valide(ch):
16     if ch.isdecimal() and len(ch)==10 and  2000 <= int(ch[:4]) <= 2023 and  1<= int(ch[4:6]) <= 12:
17         return True
18     else:
19         return False
20 #-----
21 # Calcul de nombre de mois d'ancienneté
22 def Anciennete(ch):
23     a=int(ch[:4])
24     m=int(ch[4:6])
25     mm=5
26     aa=2023
27     an = aa-a;
28     if (mm < m) and (an!=0):
29         an = an-1
30     if m <= mm :
31         nb = an*12+(mm-m)
32     else :
33         nb = an*12+(12-(m-mm))
34     return nb
35 #-----
36 def Bonus():
37     form.res.setText(Fidelite(form.a.text()))
38 #-----
39 app = QApplication([])
40 form = loadUi ("Interface_adhesion.ui")
41 form.show()
42 form.b.clicked.connect (Bonus)
43 app.exec_()
44
```

