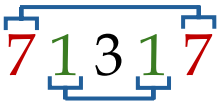
Sujet 08h : Premier Palindrome

On se propose de concevoir une interface graphique permettant de saisir un nombre N de trois chiffres au minimum, puis d'afficher s'il est premier palindrome.

- Un nombre est dit **premier palindrome** s'il est à la fois premier et palindrome.
- Un nombre est dit **premier** s'il est divisible seulement par 1 et lui-même.
- Un nombre est dit **palindrome** s'il est symétrique, c'est-à-dire qu'il se lit de la même manière de droite à gauche et de gauche à droite.

Exemples :

Pour **N = 71317** : 71317 est à la fois un nombre premier et un nombre palindrome.



Le programme affiche : **71317 est premier palindrome**

Pour **N = 232** : 232 n'est pas un nombre premier.

Le programme affiche : **232 n'est pas premier palindrome**

Pour **N = 137** : 137 est premier, mais il n'est pas un nombre palindrome.

Le programme affiche : **137 n'est pas premier palindrome**

Pour **N = 2514** : 2514 n'est ni premier ni palindrome.

Le programme affiche : **2514 n'est pas premier palindrome**

Travail demandé

1) Créer l'interface graphique illustrée dans la figure **Fig-1** et l'enregistrer sous le nom **Interface**. Cette interface contient les éléments suivants :

- Un label contenant le texte : "**Premier Palindrome**".
- Un label contenant le texte : "**N =** ".
- Une zone de saisie pour la saisie d'un nombre **N**.
- Un bouton intitulé "**Vérifier**".
- Un label dédié à l'afficher.

Fig-1

2) Créer un programme python et l'enregistrer sous le nom **PrePal**, dans lequel on demande :

a) de développer une fonction **Premier(A)** qui permet de tester si un entier strictement positif **A** est premier ou non.

b) d'implémenter l'algorithme suivant de la fonction **Palindrome** qui permet de vérifier si une chaine de caractères **CH** est palindrome ou non.

```
Fonction Palindrome(CH: Chaîne):Booléen
Début
  i ← 0
  j ← Long(CH) - 1
  TantQue (i<j) et (CH[i]=CH[j]) Faire
    i ← i+1
    j ← j-1
  Fin TantQue
  Retourner i ≥ j
Fin
```

T.D.O.L

Objet	Type/Nature
i, j	Entier

c) de développer une fonction **Verif(N)** qui permet de vérifier si un nombre **N** est un nombre **premier palindrome** ou non, en utilisant les fonctions **Premier** et **Palindrome** précédentes.

d) de développer un module **Play,** qui s'exécute à la suite au clic sur le bouton "**Vérifier**", permettant :

- de récupérer la valeur du nombre **N** saisi, de s'assurer de sa validité et d'afficher, le cas échéant, le message adéquat via le label dédié à l'affichage, comme illustré dans la figure **Fig-2**.
- d'exploiter la fonction **Verif** afin d'afficher le message adéquat via le label dédié à l'affichage, comme illustré dans les figures **Fig-3**, **Fig-4** et **Fig-5**.

e) d'exploiter l'annexe présentée ci-après tout en apportant les modifications nécessaires à l'intégration de l'interface graphique **Interface**.

Fig-2

Fig-3

Fig-4

Fig-5

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
1. Création de l'interface **Interface.**	3
2. Création du programme **PrePal**.	17
a) Développement de la fonction **Premier**	4
b) Implémentation de la fonction **Palindrome**	4
c) Développement de la fonction **Verif**	1.5
d) Développement du module **Play**.	6
e) Exploitation de l'annexe.	1.5

Sujet 09h30 : Diviseurs Unitaires

On se propose de concevoir une interface graphique permettant de saisir un entier naturel **N** composé de trois chiffres et d'afficher **ses diviseurs unitaires** s'ils existent.

On dit qu'un entier **A est un diviseur unitaire** d'un entier **N** si et seulement s'il existe un entier **B** tels que :

- **N** = **A** * **B**
- A est composé d'un seul chiffre différent de 1.
- **A** et **B** sont premiers entre eux. Deux entiers sont dits premiers entre eux si leur plus grand commun diviseur (PGCD) est égal à 1.

Exemples :

1) Pour **N = 252**

252 = 4 * 63 4 et 63 sont premiers entre eux PGCD (4,63) = 1	252 = 7 * 36 7 et 36 sont premiers entre eux PGCD (7,36)= 1	252 = 9 * 28 9 et 28 sont premiers entre eux PGCD (9,28) = 1

Le programme affiche : **Les diviseurs unitaires de 252 sont : 4, 7, 9**

2) Pour **N = 901**

N n'a aucun diviseur formé d'un seul chiffre.

Le programme affiche : **901 ne possède aucun diviseur unitaire**

3) Pour **N = 999**

999 = 3 * 333 3 et 333 ne sont pas premiers entre eux PGCD (3,333) = 3 ≠ 1	999 = 9 * 111 9 et 111 ne sont pas premiers entre eux PGCD (9,111) = 3 ≠ 1

Le programme affiche : **999 ne possède aucun diviseur unitaire**

Travail demandé

1. Créer l'interface graphique illustrée dans la figure **Fig-1** et l'enregistrer sous le nom **Interface**. Cette interface contient les éléments suivants :
   - Un label contenant le texte : "**Diviseurs Unitaires**"
   - Un label contenant le texte : "**N =**".
   - Une zone de saisie pour la saisie d'un entier **N**.
   - Un bouton intitulé "**Afficher**".
   - Un label dédié à l'affichage.

Fig-1

2. Créer un programme python et l'enregistrer sous le nom **DivUnit**, dans lequel on demande :
   a. d'implémenter l'algorithme suivant de la fonction **PGCD** qui permet de déterminer le plus grand commun diviseur de deux entiers **A** et **B**.

```
Fonction PGCD(A, B: Entier):Entier
DEBUT
  TantQue (B ≠ 0) Faire
    R ← A MOD B
    A ← B
    B ← R
  Fin TantQue
  Retourner A
FIN
```

T.D.O.L

Objet	*Type/Nature*
R	Entier

b. de développer la fonction **Verif(N, A)** qui permet de vérifier si l'entier **A** est un **diviseur unitaire** de l'entier **N** en exploitation la fonction **PGCD** précédente.
c. de développer la fonction **Unitaire(N)** qui permet de déterminer les **diviseurs unitaires de N** s'ils existent en exploitant la fonction **Verif** précédente.
d. de développer un module **Play** qui s'exécute à la suite au clic sur le bouton "**Afficher**" permettant :
    i. de récupérer la valeur de l'entier **N** saisi , de s'assurer de sa validité et d'afficher, le cas échéant, le message adéquat via le label dédié à l'affichage, comme illustré dans la figure **Fig-2**.
    ii. d'exploiter la fonction **Unitaire** précédente afin d'afficher le message adéquat via le label dédié à l'affichage, comme illustré dans les figures **Fig-3** et **Fig-4**.
e. d'exploiter l'annexe présentée ci-après tout en apportant les modifications nécessaires à l'intégration de l'interface graphique **Interface**.

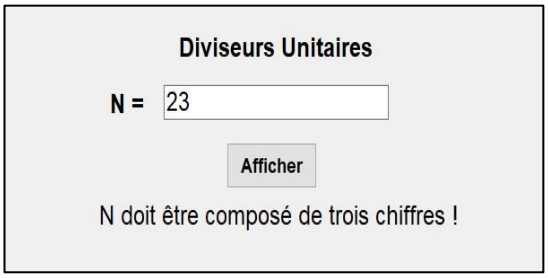


Fig-2

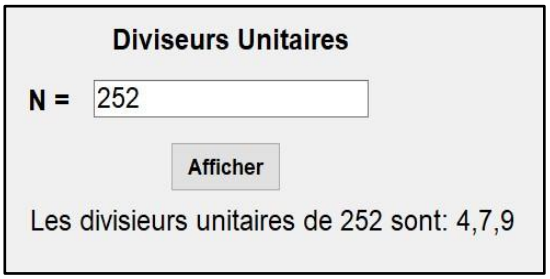


Fig-3

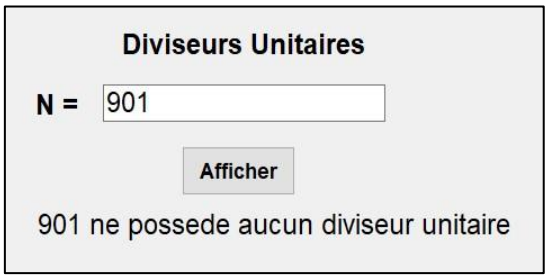


Fig-4

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
1) Création de l'interface **Interface.**	**3**
2) Création du programme **DivUnit**.	**17**
a) Implémentation de la fonction **PGCD**	**4**
b) Développement de la fonction **Verif**	**3**
c) Développement de la fonction **Unitaire**	**4**
d) Développement du module **Play**.	**4.5**
e) Exploitation de l'annexe.	**1.5**

Sujet 11h : Nombres ronds

On se propose de concevoir une interface graphique permettant de saisir deux entiers **A** et **B** tels que (10 ≤ **A** ≤ 30) et (A < **B** ≤ 99), puis d'afficher tous les nombres ronds compris entre **A** et **B**. Un nombre est dit **rond** si sa conversion à la base 2 (base binaire) contient autant de 1 que de 0 (le nombre des 0 égal à celui des 1).

Exemples :

Pour **A = 20 et B = 40** :

Le programme affiche : **Les nombres ronds sont : 35-37-38**

En effet,

- la conversion binaire de $35_{(10)}$ vaut $100011_{(2)}$ qui contient autant de 1 que de 0.
- la conversion binaire de $37_{(10)}$ vaut $100101_{(2)}$ qui contient autant de 1 que de 0.
- la conversion binaire de $38_{(10)}$ vaut $100110_{(2)}$ qui contient autant de 1 que de 0.

Pour **A = 15 et B = 30** :

Il n'existe aucun nombre rond compris entre **15** et **30**, d'où :

Le programme affiche : **Aucun nombre rond**

Travail demandé

1. Créer l'interface graphique illustrée dans la figure **Fig-1** et l'enregistrer sous le nom **Interface**. Cette interface contient les éléments suivants :
   - Un label contenant le texte : "**Nombres ronds**".
   - Un label contenant le texte : "**A =** ".
   - Une zone de saisie pour la saisie d'un entier **A**.
   - Un label contenant le texte : "**B =** ".
   - Une zone de saisie pour la saisie d'un entier **B**.
   - Un bouton intitulé "**Afficher**".
   - Un label dédié à l'affichage.

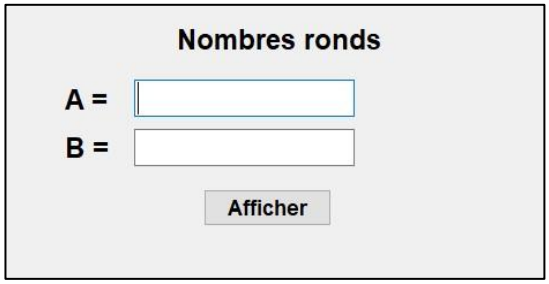


Fig-1

2. Créer un programme python et l'enregistrer sous le nom **NombreRond**, dans lequel on demande :
   a. d'implémenter l'algorithme suivant de la fonction **Conv_binaire** qui permet de convertir un entier **N** en binaire.

```
Fonction Conv_binaire(N: Entier):Chaîne
Début
  Si (N=0) Alors
    ch ← "0"
  Sinon
    ch ← ""
    TantQue N ≠ 0 Faire
      r ← N mod 2
      N ← N Div 2
      ch ← Convch (r) + ch
    Fin TantQue
  Fin Si
  Retourner ch
FIN
```

T.D.O.L

Objet	Type/Nature
ch	Chaîne de caractères
r	Entier

   b. de développer une fonction **Verif(N)** qui permet de vérifier si un entier **N** est rond, en utilisant la fonction **Conv_binaire** précédente.

c. de développer une fonction **Ronds(A, B)** pour former une chaine de caractères contenant les nombres ronds, de l'intervalle [A..B], séparés par le caractère "-", en utilisant la fonction **Verif**.

d. de développer un module **Play**, qui s'exécute suite au clic sur le bouton "**Afficher**", permettant :

   i. de récupérer les valeurs saisies des entiers **A** et **B**, de s'assurer de leur validité et d'afficher, le cas échéant, le message adéquat via le label dédié à l'affichage, comme illustré dans les figures **Fig-2** et **Fig-3**.

   ii. d'exploiter la fonction **Ronds** afin d'afficher le message adéquat via le label dédié à l'affichage, comme illustré dans les figures **Fig-4** et **Fig-5**.

e. d'exploiter l'annexe présentée ci-après tout en apportant les modifications nécessaires à l'intégration de l'interface graphique **Interface**.

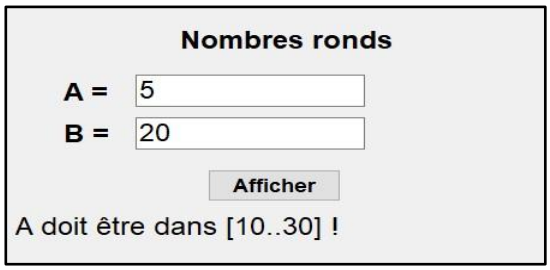


Fig-2

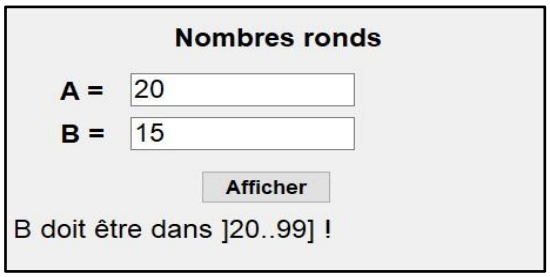


Fig-3

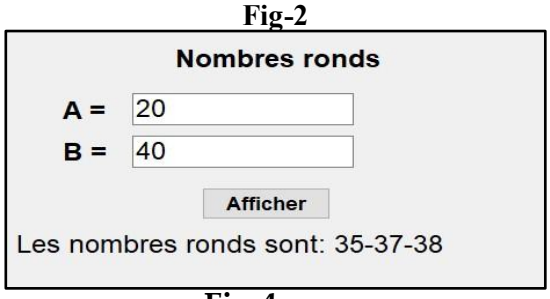


Fig-4

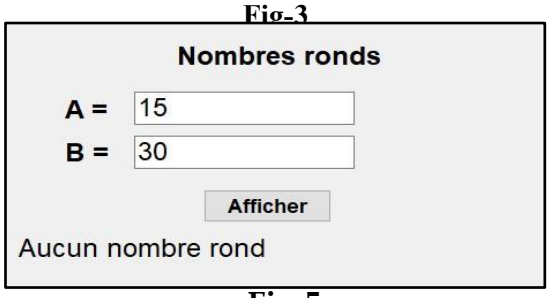


Fig-5

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
1) Création de l'interface **Interface.**	**3**
2) Création du programme **Nombre_Rond**.	**17**
a) Implémentation de la fonction **Conv_binaire**	**4**
b) Développement de la fonction **Verif**	**3**
c) Développement de la fonction **Ronds**	**2.5**
d) Développement du module **Play**.	**6**
e) Exploitation de l'annexe.	**1.5**

Sujet 13h : Nombres d'Armstrong

On se propose de concevoir une interface graphique permettant de saisir deux entiers **A** et **B** tels que **(10 ≤ A ≤ 1000)** et **(A < B ≤ 10000)**, puis d'afficher tous les nombres d'Armstrong compris entre **A** et **B**. Un nombre d'Armstrong **N** est un entier positif égal à la somme de ses propres chiffres, élevés chacun à la puissance du nombre de chiffres de **N**.

Exemple :

Pour **A** = 200 et **B** = 1700, le programme affiche : 370, 371, 407, 1634 En effet :

- $370 = 3^3 + 7^3 + 0^3$
- $371 = 3^3 + 7^3 + 1^3$
- $407 = 4^3 + 0^3 + 7^3$
- $1634 = 1^4 + 6^4 + 3^4 + 4^4$

Pour **A** = 500 et **B** = 1500, le programme affiche : **Aucun nombre d'Armstrong dans cet intervalle**

Travail demandé

1. Créer l'interface graphique illustrée dans la figure **Fig-1** et l'enregistrer sous le nom **Interface**. Cette interface contient les éléments suivants :
   - Un label contenant le texte : "**Nombres d'Armstrong**".
   - Un label contenant le texte : "**A =** ".
   - Une zone de saisie pour la saisie d'un entier **A**.
   - Un label contenant le texte : "**B =** ".
   - Une zone de saisie pour la saisie d'un entier **B**.
   - Un bouton intitulé "**Afficher**".
   - Un label dédié à l'affichage.

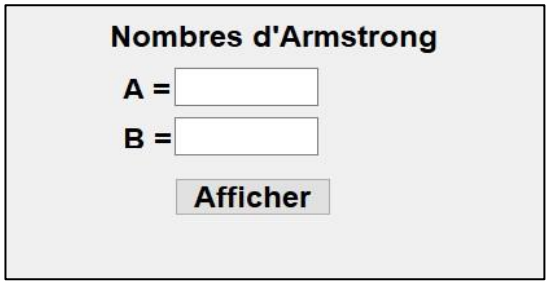


Fig-1

2. Créer un programme python et l'enregistrer sous le nom **Armstrong**, dans lequel on demande :
   a. d'implémenter l'algorithme suivant de la fonction **PUISS** qui permet de retourner **N** à la puissance **P** (N^P).

```
Fonction PUISS (N, P : Entier) : Entier
DEBUT
  A ← 1
  Pour i de 1 à P Faire
    A ← A * N
  Fin Pour
  Retourner A
FIN
```

T.D.O.L

Objet	*Type/Nature*
A	**Entier**
i	**Entier**

   b. de développer une fonction **SommePuiss(N)** qui retourne la somme des chiffres de N élevé chacun à la puissance qui correspond aux nombres de chiffres de l'entier N en exploitant la fonction **PUISS.**
   c. de développer un module **Play** qui s'exécute à la suite au clic sur le bouton "**Afficher**" permettant :
      i. de récupérer les valeurs saisies des entiers **A** et **B**, de s'assurer de leur validité et d'afficher, le cas échéant, le message adéquat via le label dédié à l'affichage, comme illustré dans les figures **Fig-2** et **Fig-3**.
      ii. d'exploiter la fonction **SommePuiss** afin d'afficher le message adéquat via le label dédié à l'affichage, comme illustré dans les figures **Fig-4** et **Fig-5**.

d. d'exploiter l'annexe présentée ci-après tout en apportant les modifications nécessaires à l'intégration de l'interface graphique **Interface**.

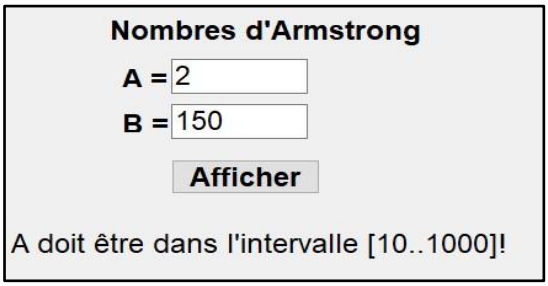


Fig-2

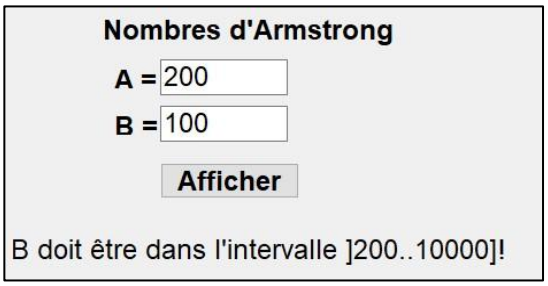


Fig-3

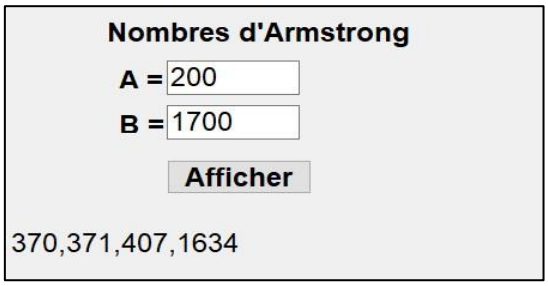


Fig-4

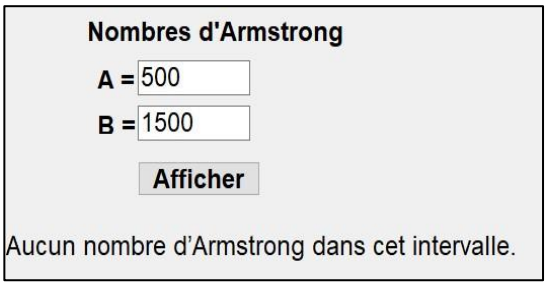


Fig-5

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
1) Création de l'interface **Interface.**	**3**
2) Création du programme	**17**
a) Implémentation de la fonction **PUISS**	**4**
b) Développement de la fonction **SommePuiss**	**5**
c) Développement du module **Play**	**6.5**
d) Exploitation de l'annexe.	**1.5**

Sujet 14h30 : Mots ordonnés

On se propose de concevoir une interface graphique permettant de saisir une chaîne de caractères **Ch** et d'afficher sans redondance ses mots qui sont triés selon l'ordre alphabétique, séparés par le caractère "**-**". Un mot est trié selon l'ordre alphabétique si tous ses caractères sont ordonnés en ordre croissant des lettres alphabétiques.

Exemples :

Pour **Ch**="**le lion mange la viande.**", le programme affiche : **Aucun mot trié dans la chaine**

Pour **Ch**="**il est gentil.**", le programme affiche : **il-est**

En effet, les caractères des deux mots "**il**" et "**est**" sont triés en ordre croissant alphabétiquement.

Pour **Ch**="**il est fou.**", le programme affiche : **il-est-fou**

En effet, les caractères des trois mots "**il**", "**est**" et "**fou**" sont triés en ordre croissant.

Pour **Ch**="**il est gentil et il est fou.**" le programme affiche : **il-est-et-fou**

En effet, les caractères des mots "**il**", "**est**", "**et**" et "**fou**" sont triés en ordre croissant alphabétiquement et les mots "**il**" et "**est**" sont affichés une seul fois chacun.

Travail demandé

1. Créer l'interface graphique illustrée dans la figure **Fig-1** et l'enregistrer sous le nom **Interface**. Cette interface contient les éléments suivants :
   - Un label contenant le texte : "**Mots ordonnés**".
   - Un label contenant le texte : "**La chaine :** ". Une zone de saisie pour la saisie de la chaine de caractères **Ch**.
   - Un bouton intitulé "**Afficher**".
   - Un label dédié à l'affichage.

Fig-1

2. Créer un programme python et l'enregistrer sous le nom **MotOrd**, dans lequel on demande :
   a. d'implémenter l'algorithme suivant de la fonction **Ordonner (Mot)** qui permet de vérifier si la chaîne de caractères **Mot** est triée en ordre croissant des lettres alphabétique ou non.

```
Fonction Ordonner(Mot: Chaîne):Booléen
DEBUT
 i ← 1
 TantQue i<Long(Mot) et Mot[i-1]≤Mot[i] Faire
  i ← i+1
 Fin Tant que
 Retourner i=Long(Mot)
FIN
```

T.D.O.L

Objet	*Type/Nature*
i	Entier

   b. de développer une fonction **Traitement (ch)** qui permet d'exploiter la fonction **Ordonner** et de retourner les mots de la chaine de caractères **ch** qui sont ordonnés sans redondance séparés par le caractère "**-**".
   c. de développer un module **Play,** qui s'exécute à la suite du clic sur le bouton "**Afficher**", permettant :

i. de récupérer la chaine de caractères saisie, qui doit être une chaine de caractères non vide, se termine par le caractère ".", formée au maximum par 30 caractères (lettres minuscules et espaces uniquement) et deux mots consécutifs sont séparés par un seul espace. Si au moins une contrainte n'est pas respectée, afficher le message "***Veuillez saisir une chaine valide !***" comme illustré dans les figures **Fig-2** et **Fig-3**.

ii. d'exploiter la fonction **Traitement** afin d'afficher sans redondance les mots triés via le label dédié à l'affichage, comme illustré dans les figures **Fig-4**, **Fig-5**, **Fig-6** et **Fig-7**.

d. d'exploiter l'annexe présentée ci-après tout en apportant les modifications nécessaires à l'intégration de l'interface graphique **Interface**.

Fig-2

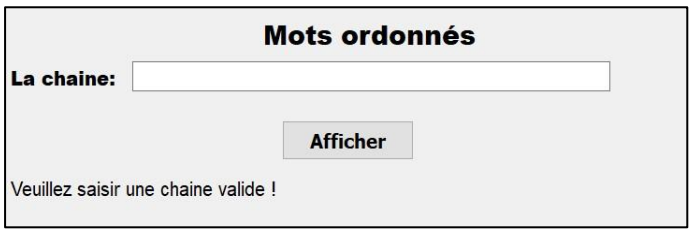


Fig-3

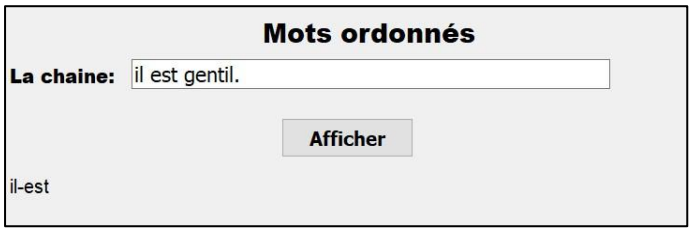


Fig-4

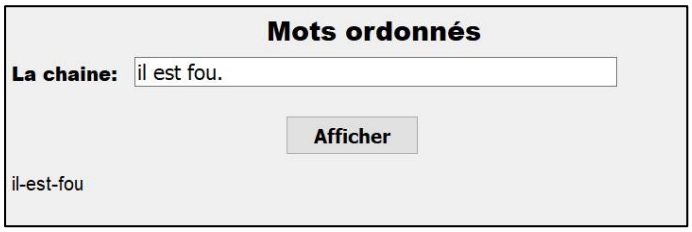


Fig-5

Fig-6

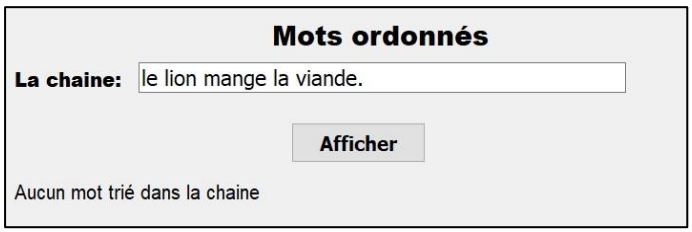


Fig-7

Annexe

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import Qpplication
……
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
1) Création de l'interface **Interface.**	**3**
2) Création du programme **MotOrd**	**17**
a) Implémentation de la fonction **Ordonner**	**4**
b) Développement de la fonction **Traitement**	**5.5**
c) Développement du module **Play**	**6**
d) Exploitation de l'annexe.	**1.5**