ISN 8 : Aide à la réalisation du jeu du pendu

- **Opérations et fonctions utiles sur les chaînes de caractères** :
  - **La longueur** : la fonction **len** permet de récupérer la longueur d'une chaîne de caractères.
  - La **concaténation** : opération qui consiste à construire une chaîne de caractères en mettant bout à bout d'autres chaînes de caractères.

```
>>> 'Hello '+"world!"
'Hello world!'
>>> x = 'Hello '
>>> y = "world!"
>>> z = x + y
>>> z
'Hello world!'
```

  - **Répétition** : opération pour construire des chaines ayant des structures répétitives

```
>>> 'Hello world!' * 3
'Hello world!Hello world!Hello world!'
```

  - **Egalité**

```
>>> "abc" == 'abc'
True
>>> "abc" != "ABC"
True
```

- L'utilisation de caractères spéciaux doit être précédé d'un backslash :

```
fich = open("d:\mots.txt","r")
liste=fich.readlines()
print(liste)
print(liste[1])
num = random.randint(1,881)
fich.close()
```

```
'personnage\n', 'coucher\n', 'interet\n', 'dernier\n',
 'paysage\n', 'achever\n', 'attendre\n', 'enfoncer\n',
'chanter\n', 'poitrine\n', 'pencher']
detruire

>>>
```

Le premier *print* affiche toute la liste avec les retours à la ligne (« \n ») sauf le dernier mot(indication de fin de fichier).
Le deuxième *print* affiche un mot du fichier et passe à la ligne (" detruire\n").

- Python ne permet pas de changer un caractère dans une chaîne.
  Par exemple, changer le x de penxu par un d ne fonctionne pas avec le code suivant :

```
mot= "penxu"
mot[3]= "d"
```

On dit les chaînes de caractères ne sont pas **mutables**.

Pour contourner cette difficulté, on peut utiliser l'une des deux techniques suivantes (pages suivantes) :

Technique 1 : voir ISN 3, utilisation de « **sous–chaînes** »

Technique 2 : **construction séquentielle** d'un nouveau mot

Technique 1 : voir ISN 3, utilisation de « **sous–chaînes** »

```
mot= "penxu"
print(len(mot))
# suite de 3 instructions équivalentes : affichage du caractère de rang 3 jusqu'à la fin du mot choisi, ici penxu
print(mot[3:])
print(mot[3:len(mot)])
print(mot[3:5])
print() #provoque un saut de ligne
#suite de 2 instructions équivalentes : # affichage du caractère de rang 0 jusqu'au caractère de rang 2
print(mot[:3])
print(mot[0:3])
print("\n") #provoque deux sauts de lignes (un avec la fonction print et un avec le caractère spécial de passage à la ligne \n
# concaténation pour changer le x de penxu par un d
mot=mot[:3]+"d"+ mot[4:]
print(mot)
print("\n")
# concaténation équivalente
mot= "penxu"
mot=mot[0:3]+"d"+ mot[4:len(mot)]
print(mot)
```

Fichier 81.py

qui donne (programme 81.py) :

```
5
xu
xu
xu

pen
pen


pendu


pendu
```

Technique 2 : construction séquentielle d'un nouveau mot

```
mot="penxu"
print(len(mot),"\n")
nouveaumot=""  #la variable nouveaumot ne contient rien
               # mais elle est de type chaîne dont permet une opération de concaténation
               #La longueur de nouveaumot est 0
for i in range (len(mot)) :
    print(i)
    if i != 3 :
        nouveaumot=nouveaumot + mot[i]
        print(nouveaumot+"\n")
    else:
        nouveaumot=nouveaumot + "d"
mot=nouveaumot
print(mot)
```

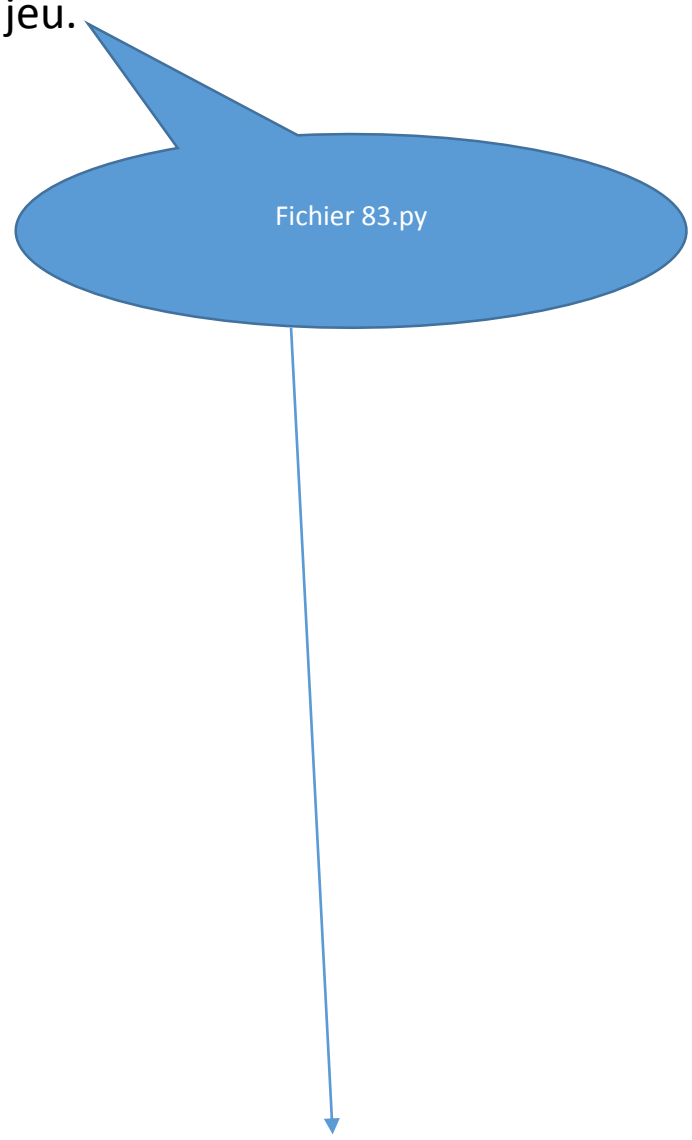
Fichier 82.py

qui donne (programme 82.py) :

```
5

0
p

1
pe

2
pen

3
4
pendu

pendu
```

Jalons pour le jeu du pendu

1. Ouvrir un fichier mots.txt contenant une liste de mots
2. Choisir un mot pris au hasard du fichier mots.txt
3. Savoir retirer le « \n » du mot choisi
4. Savoir remplacer une lettre du mot choisi par une autre lettre à une position bien spécifique.
   Par exemple, si « tort » est le mot choisi, remplacer le premier t par un p et afficher le nouveau mot « port »
5. Finir le jeu.



```python
import random
fich = open("d:\mots.txt","r")
liste=fich.readlines()
indice = random.randint(0,len(liste)-1)
mot=liste[indice]
fich.close()
mot=mot[0:len(mot)-1]
print(mot)
solution="?"*len(mot)
coup=0
while solution != mot :
    lettre=input("Proposer une lettre")
    coup=coup+1
    for i in range(len(mot)):
        if lettre == mot[i] :
            solution=solution[:i]+lettre+solution[i+1:]
    print("Après proposition de la lettre "+lettre + " : " + solution )
print("Gagné en ", coup, "essais.")
```

```
remplir
Après proposition de la lettre r : r?????r
Après proposition de la lettre u : r?????r
Après proposition de la lettre e : re????r
Après proposition de la lettre m : rem???r
Après proposition de la lettre p : remp??r
Après proposition de la lettre l : rempl?r
Après proposition de la lettre i : remplir
Gagné en  7 essais.
```

Optimisation possible ?
Le code contient un « bug » qui se produit rarement : saurez-vous le trouver ?