Autour du jeu «le pendu»

Auteur : DELROT Sylvain

Auteur : HOUNKPATIN Bonaventure ,

Auteur :MORARD – LAVALLETTE Laurence

Auteur : SAUVERGEAT Magali

Table des matières

1. Introduction

Pré-requis utilisés :

- Bases de Python (boucles, listes, dictionnaires)
- Bases HTML et CSS (générer un tableau)

Le projet se situe à la fin du premier trimestre et s'étendra jusqu'au deuxième trimestre.

Il s'étale sur plusieurs séances et des éléments seront à finir en travail à la maison.

Il fait intervenir les parties du programme sur :

- les représentations de données (types construits, tableau indexé, dictionnaire par clés)
- le traitement des données en tables (recherche dans une table)
- Indexation de tables
- les modalités de l'interaction entre l'homme et la machine (code HTML)

L'objectif est de partir d'une base de Pendu en mode texte, corrigé en classe, et de faire un mini projet de 3 élèves, chaque élève aura une partie différente à coder :

Élève 1 : Coder le tirage d'un mot aléatoire dans un fichier txt, et coder le mode démo qui proposera des lettres

Élève 2 : Coder un affichage graphique de l'évolution du pendu, les coordonnées de chaque partie du pendu seront stockées dans une liste de coordonnées

Élève 3 : Générer une page HTML d'aide aux joueurs de pendu où la fréquence des lettres en français sera donnée dans un tableau, coder le comptage des lettres dans un fichier texte, pour en extraire un dictionnaire de fréquence des lettres, coder la probabilité d'apparition d'une lettre dans un fichier texte, pour en extraire un dictionnaire de probabilité d'apparition d'une lettre dans un mot.

2. Durée et début

Durée approximative :- > 5 semaines

Séance Cours / TD : de 2H TP + 1h de Cours

Séance Projet : 6H

Séance Finale : de 2H à 3H

Ces 5 semaines débutent vers début décembre, soit le début du Trimestre 2.

3. Séance de cours /TD : le pendu en mode texte

But : Créer des fonctions en Python permettant de découvrir le mot caché du pendu.

Le professeur demande aux élèves de créer plusieurs fonctions dont les noms sont donnés ainsi que les rôles. Les noms des fonctions sont imposés pour plus de facilité et de lisibilité du code.
Remarque : les chaînes de caractères sont non modifiables lettre par lettre en python.On obligera l'élève à utiliser des listes de lettres et non des chaînes de caractères. Cela force aussi les élèves à écrire à la main certaines fonctions simples (présence d'une lettre, etc) et s'entraîner avec les listes sans avoir recours à certains raccourci python (« in »)

Ce TP pourra se faire sur 2 niveaux progressifs :

Cette version part d'un mot à deviner fixe.

- **genere_tirets,** cette fonction prend en paramètre un nombre et génère une liste de tirets correspondant à la taille du mot cherché.
- **lettre_presente**, cette fonction prend en paramètre le mot caché sous forme de liste ainsi qu'une lettre proposée. Cette fonction doit tester si la lettre est présente dans le mot et elle doit renvoyer un booléen True ou False.
- **devoiler_lettre**, cette fonction prend en paramètre le mot caché sous forme de liste, la liste contenant des tirets, ainsi qu'une lettre proposée. Cette fonction doit remplacer la lettre présente dans le mot et laisser des tirets pour les lettres inconnues.
- **teste_gagne,** cette fonction prend en paramètre le mot caché sous forme de liste, la proposition du joueur sous forme de liste. Cette fonction doit tester si le mot caché est identique à la proposition du joueur, elle doit renvoyer un booléen True ou False.
- **prog_principal,** Cette fonction appelle les autres fonctions dans une boucle tantque le mot n'est pas deviné ou le nombre d'erreurs maximum n'est pas atteint.

On pourra fixer le nom des variables via ce début de code :

```
erreurmax=12
motcache= list("octogone")
lettresprop=[]
nberreur=0
devine=genere_tirets(len(motcache))
```

Le code sera corrigé en cours et un programme python de correction sera donné. Des groupes de 3 élèves seront formés pour partir en projet.

L'évaluation sera la moyenne d'une note personnelle et d'une note d'équipe.

4. Élève 1 : Tirer au hasard le mot et mode démo via le tableau de fréquence

But : Créer une intelligence artificielle, l'ordinateur doit trouver le mot caché en le moins d'étapes possibles.
Le travail de l'élève s'articule autour de 4 fonctions utiles à l'enrichissement du pendu vu en cours
Les différentes fonctions à créer sont les suivantes :

Les deux premières fonctions ont pour but de choisir aléatoirement un mot dans un fichier texte.

- **choisi_mot,** cette fonction prend en paramètre la taille maximum du mot. Cette fonction lit le fichier .txt de mots et l'enregistre dans un tableau liste_mot puis choisi au hasard un mot via la fonction choice du module random. Ce mot ne devra pas être d'une taille supérieure à la taille demandée. Cette fonction renvoie le mot choisi.
- **normalise,** cette fonction prend en paramètre un mot avec ou sans accents et retourne un mot sans accent. Cette fonction doit permettre de normaliser les mots accentués du fichier.txt en mots non accentués.

Les deux fonctions suivantes doivent gérer une version démo.

- **demo_v1 :** qui prend en paramètre la liste des lettres déjà proposées, tire une lettre au hasard via la fonction choice du module random, teste si elle ne fait pas partie des lettres proposées.
- **demo_v2**

L'élève utilise un dictionnaire fixe des fréquences des lettres en français, ce dictionnaire fixe dans un premier temps sera généré dans un deuxième temps par l'élève 3.

```
frequence = {'a' : 0.083944, 'b' : 0.007669,  'c' : 0.033297, 'd' : 0.040699, 'e' :
0.145037,'f'  :  0.012109,  'g'  :  0.009495,  'h'  :  0.007973,  'i'  :  0.081828,  'j'  :
0.006377,  'k'  :  0.000638,  'l'  :  0.058405,'m'  :  0.029355,  'n'  :  0.075570,  'o'  :
0.053669,  'p'  :  0.032087,  'q'  :  0.012613,  'r'  :  0.070209,  's'  :  0.080091,'t'  :
0.074775,  'u'  :  0.059808,  'v'  :  0.015791,  'w'  :  0.000067,  'x'  :  0.004098,  'y'  :
0.003155, 'z' : 0.001240}
```

- **demo_v2 :** qui prend en paramètre la liste des lettres déjà proposées, ainsi qu'un dictionnaire des fréquences des lettres de l'alphabet. Cette fonction parcourt le dictionnaire pour trouver la lettre de fréquence maximum du dictionnaire qui ne fait pas partie des lettres déjà proposées.

Documents fournis à l'élève:

Un exemple lisant un fichier mot à mot. Un dictionnaire python des fréquences des lettres de la langue française trouvée sur http://www.apprendre-en-ligne.net/crypto/stat/francais.html

Évaluation du travail individuel :

Le code réalisé correspond à ces critères :

Fonction permettant de choisir un mot d'une taille donnée dans le fichier correcte

Gestion des accents et des lettres spéciales

A choix des lettres aléatoire :

Utilisation d'une liste de lettres

Gestion correcte des lettres déjà proposées (mise à jour de la liste)

Utilisation de randint() ou choice()

B choix des lettres en fonction des fréquences

Recherche de la lettre non proposée de fréquence maximale

Script rendu fonctionnel sans bug ou plantage

Un compte rendu montrant les tests des fonctions et les test des cas d'erreurs est rendu.

5. Élève 2: Dessiner un pendu via Turtle

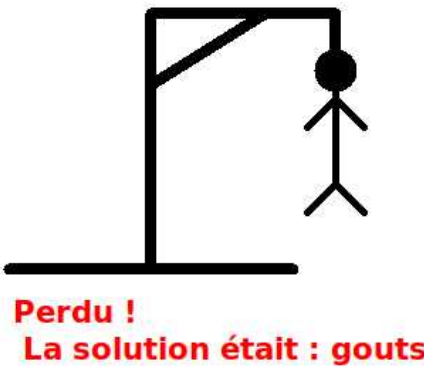
But : Dessiner le pendu grâce au module Turtle de Python.

Le professeur demande à l'élève de créer un programme permettant de dessiner un pendu, il peut montrer aux élèves le résultat escompté, l'élève doit « découper » le pendu en 11 étapes : le bas de la potence,le mât de la potence, le haut de la potence, la barre oblique, la corde, la tête, le corps, le bras droit, le bras gauche, la jambe droite et la jambe gauche

Chaque étape sera insérée dans une liste contenant la coordonnée du début du trait et la coordonnée de fin de trait. Un traitement particulier sera réalisé pour la tête.

```
Placement = [(0,-70), (0,-100), # placement du texte
            (-150,-30),(50,-30),#plancher
             (-50,-30),(-50,150),#barre verticale
             (-50,150),(80,150),#barre horirontale haute
             (-50,100),(30,150),#barre oblique
             (80,150),(80,120),#haut corde
             (80,110),(80,110),#tete
            (80,100),(80,30), #colonne vertébrale
            (80,30),(60,10),#jambe droite
            (80,30),(100,10),#jambe gauche
            (80,90),(60,70),  #bras droit
            (80,90),(100,70) ]  #bras gauche
```

- **dessin_pendu,** cette fonction prend en paramètre l'objet permettant de dessiner, le numéro de l'étape ainsi que le tableau de placement. Cette fonction dessine le trait correspondant, avec un cas particulier pour la tête de l'étape 6.
- **dessin_texte** cette fonction prend en paramètre l'objet permettant de dessiner, le texte à écrire, la position où l'écrire, la couleur de l'écriture. Cette fonction écrit le texte à la position demandée.



Documents fournis à l'élève:

Un code d'exemple , affichant 3 traits, un texte et la saisie d'un mot.

Évaluation du travail individuel :

Le code réalisé correspond à ces critères :

Séparation correcte des instructions en rapport aux différentes parties du dessin à produire

Respect des coordonnées indiquées sur la figure

Gestion des déplacements du feutre sans trace (penup() pendown())

Écriture correcte des tests pour l'automate à état

Script rendu fonctionnel sans bug ou plantage

Un compte rendu montrant les tests des fonctions et les tests des cas d'erreurs est rendu.

6. Élève 3 : Générateur de code HTML à partir d'un fichier texte

But : réaliser un tableau en utilisant HTML et CSS à partir d'un dictionnaire de données

On donne à l'élève un dictionnaire python des fréquences des lettres alphabétiques en français.

```
frequence = {'a' : 0.083944, 'b' : 0.007669,  'c' : 0.033297, 'd' : 0.040699, 'e' : 0.145037,  ….  'z' : 0.001240}
```

Cet élève sera responsable de 3 fonctions de niveaux progressifs.

1. **gen_html()** : prend en paramètre le dictionnaire des fréquences, et génère une page HTML contenant un tableau HTML présentant ces fréquences, cette page est une aide pour le joueur du pendu.

   On peut introduire que l'analyse des fréquences des lettres permet aux navigateurs web de trouver la langue utilisée dans un texte. La fréquence des lettres peut également être utilisée dans le décodage de messages cryptés. cf :

   https://fr.wikipedia.org/wiki/Fr%C3%A9quence_d%27apparition_des_lettres_en_fran%C3%A7ais

Fréquences des caractères:

```
A : 0.083944
B : 0.007669
C : 0.033297
D : 0.040699
E : 0.145037
F : 0.012109
G : 0.009495
H : 0.007973
I : 0.081828
J : 0.006377
K : 0.000638
L : 0.058405
M : 0.029355
N : 0.075570
O : 0.053669
P : 0.032087
Q : 0.012613
R : 0.070209
S : 0.080091
T : 0.074775
U : 0.059808
V : 0.015791
W : 0.000067
X : 0.004098
Y : 0.003155
Z : 0.001240
```

Fréquences d'apparition des lettres

Lettre	Fréquence	Lettre	Fréquence
A	8.40 %	N	7.13 %
B	1.06 %	O	5.26 %
C	3.03 %	P	3.01 %
D	4.18 %	Q	0.99 %
E	17.26 %	R	6.55 %
F	1.12 %	S	8.08 %
G	1.27 %	T	7.07 %
H	0.92 %	U	5.74 %
I	7.34 %	V	1.32 %
J	0.31 %	W	0.04 %
K	0.05 %	X	0.45 %
L	6.01 %	Y	0.30 %
M	2.96 %	Z	0.12 %

2. **lit_frequence(),** cette fonction lit un fichier contenant 26 lignes présentant « la lettre : la fréquence » le dictionnaire de fréquence sera construit à partir de ces informations. La fonction retourne ce dictionnaire. ***(cette étape peut être enlevée)***
3. **calcul_frequence()** cette fonction lit un fichier un ensemble de mots en français. La fonction compte le nombre de lettres et le nombre de chacune des lettres de l'alphabet. Puis, la fonction remplit le dictionnaire des fréquences avec ces valeurs. (i:nb de a / nb de lettres)
4. **calcul_probabilité()** cette fonction lit un fichier contenant un ensemble de mots en français. La fonction compte le nombre de mots, pour chaque mot il enlève les lettres en double via la fonction **set.** Puis il compte pour chaque lettre la somme des apparitions dans les mots sans doublons et le nombre de mots. Puis, la fonction remplit le dictionnaire des probabilités d'apparition d'une lettre dans un mot. Exemple : si la lettre « m » apparaît dans 2 mots sur 10 même si elle est répétée plusieurs fois dans ces mots la probabilité est de 2/10. (ie: nb de mots contenant la lettre/ nb de mots)

Critique et réflexion pour l'élève 3:

Le travail de l'élève 3 est un assez éloigné du jeu « le pendu », cependant, on peut argumenter que s'il arrive à finir la fonction **calcul_probabilité(),** il améliore grandement la version démo du candidat 1.

Puisque cette fonction de démonstration s'appuie sur la fréquence des lettres et non sur la probabilité des lettres dans un mot.

Probabilité d'apparition d'une lettre dans un mot

Lettre	% de Probabilité	Lettre	% de Probabilité
a	50.84	n	47.10
b	12.04	o	40.79
c	29.57	p	20.68
d	18.34	q	4.96
e	82.12	r	55.76
f	9.81	s	42.36
g	13.16	t	47.57
h	9.36	u	32.85
i	54.69	v	10.95
j	1.89	w	0.31
k	0.88	x	3.46
l	29.11	y	2.72
m	22.27	z	0.80

DIU-Bloc1

Documents fournis à l'élève:

Un exemple écrivant du texte dans un fichier et un exemple lisant un fichier caractères par caractères. Un dictionnaire python des fréquences des lettres de la langue française trouvée sur http://www.apprendre-en-ligne.net/crypto/stat/francais.html

Évaluation du travail individuel :

Le code réalisé correspond à ces critères :

Code python correct pour l'ouverture/fermeture du fichier en sortie formatée

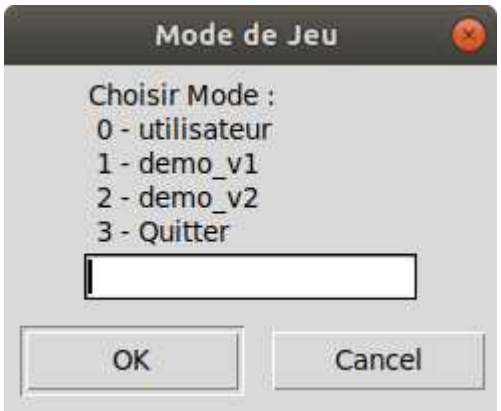
Utilisation correcte de la mise en forme avec une feuille de style CSS fournie (mot clé 'class' avec le bon nom)

Fichier HTML produit lisible sous un navigateur courant

Stockage des fréquences calculées dans un dictionnaire

Script rendu fonctionnel sans bug ou plantage

Un compte rendu montrant les tests des fonctions et les test des cas d'erreurs est rendu

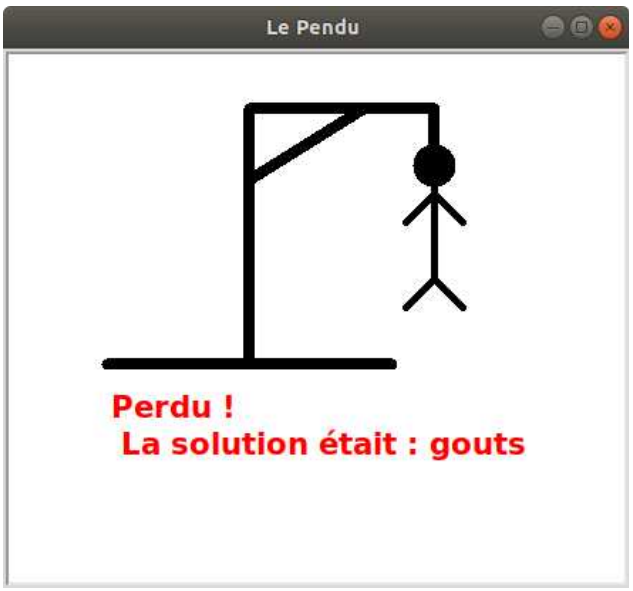


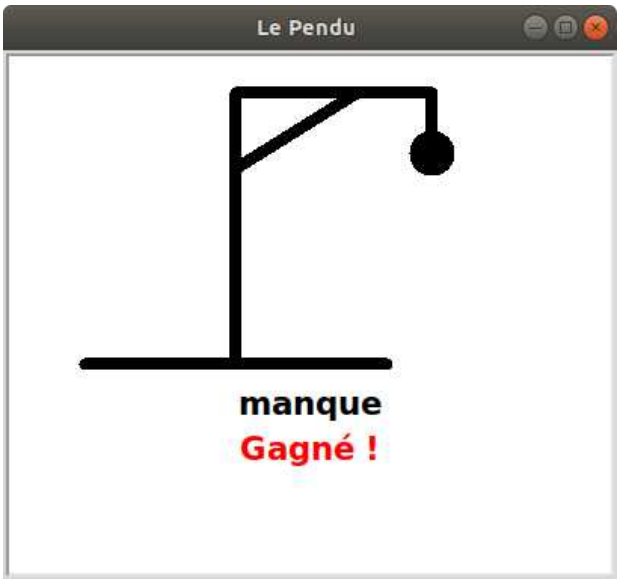
7. Séance Finale : Intégration des 3 modules

But : Finaliser l'ensemble du programme

En groupe de 3, les élèves devront mettre leur travail en commun, et également ils devront faire un menu permettant de demander à l'utilisateur, s'il veut jouer ou s'il veut que l'ordinateur réalise une démonstration, puis le jeu se déroule, le pendu se construit à chaque mauvaise réponse du joueur ou de l'ordinateur.

S'il reste du temps, ils pourront envisager de proposer des niveaux de jeu, avec des mots plus ou moins longs, ou contenant des lettres plus ou moins fréquentes.





Évaluation du travail de groupe :

Les 3 parties sont intégrées elles fonctionnent ensemble.

Un compte rendu montrant les tests du programme final est rendu.

8. Réflexion sur des fonctions avancées

But : Donner des objectifs et faire réfléchir les élèves en avance

Pour les parties 1 et 3 :

Réfléchir à des fonctions pour améliorer le mode démo.

- Gestion différente des voyelles et des consonnes
- Gestions des associations de lettres

Création d'un dictionnaire de dictionnaire d'association de 2 lettres

Dictionnaire de probabilité d'association de lettres

```
proba2alpha = {'a': {'a': 0.0, 'b': 0.06956904133685136, 'c': 0.156948109058927, 'd': … },
 'b': {'a': 0.06956904133685136, 'b': 0.0, 'c': 0.02321899736147757, 'd':…  },
'c': {'a': 0.156948109058927, 'b': 0.02321899736147757, 'c': 0.0, 'd': : … }, … }
```

Pour la partie 2 :

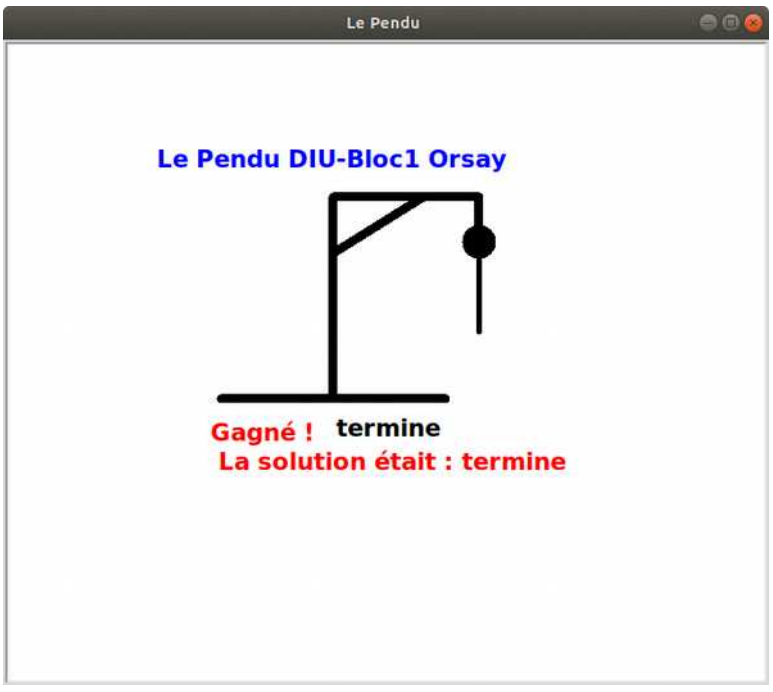
Réfléchir à l'utilisation du gestionnaire d'évènements :

```
turtle.listen()
turtle.onscreenclick(quitter, 3)
turtle.onscreenclick(reinit, 2)
turtle.onkeypress(gestion_evea, "a")
turtle.onkeypress(gestion_eveb, "b")
```

9. Conclusion et remarques

Vous trouverez joint à ce document des programmes d'exemples à fournir aux élèves pour débuter les projets et également des propositions de solutions, ces programmes ne sont pas optimums. Il serait intéressant que plusieurs personnes relisent le code et qu'ils proposent des modifications pour le rendre le plus proche d'un code d'exemple.

Si nous voulons pouvoir utiliser ce projet plusieurs fois, il serait important que les corrections finales ne soient pas données aux élèves. Sinon, les solutions risquent de se trouver sur Internet



Annexe

10. Code source de la séance 1 :

```
# Variable globale permettant de paramétrer le nombre d'érreurs possibles
ERREURMAX=12

def genere_tirets(n):
    """ permet de générer une liste de tirets représentant les lettres à trouver
        :return: ch
        :rtype: liste
    """
    ch=[]
    for i in range(n):
        ch.append("-")
    return ch

def lettre_presente(tab,l):
    """ vérifier si la lettre proposée est dans le mot à trouver
        :param tab: la liste des lettres du mot à trouver
        :type tab:  list
        :param l: la proposition dans la première position d'une liste
        :type l: list
        :rtype: booleen
    """
    for i in range(len(tab)):
        if(tab[i]==l[0]):
            return True
    return False


def devoiler_lettre(sol,prop,l):
    """ Montre la lettre dans le tableau de proposition
        :param sol: la liste des lettres du mot à trouver
        :type sol:  list
        :param prop: cette liste sera modifiée en remplaçant les tirets par la lettre
        :type prop:  list
        :param l: la proposition dans la première position d'une liste
        :type l: list
    """
    for i in range(len(sol)):
        if(sol[i]==l[0]):
            prop[i]=sol[i]

def test_gagner(sol,prop): # if(sol == prop)
    """ Vérifie si la proposition est bonne
        :param sol: la liste des lettres du mot à trouver
        :type sol:  list
        :param prop: cette liste sera modifiée en remplaçant les tirets par la lettre
        :type prop:  list
        :rtype: booleen
    """
    for i in range(len(sol)):
        if(sol[i]!=prop[i]):
            return False
    return True

def prog_principal():
```

```
    # la version 1
    # la liste contenant les lettres du mot à trouver
    motcache= list("octogone")
    # liste contenant les lettres déjà propsées
    lettresProp=[]
    # compteur du nombre d'érreurs
    nberreur=0
    # génère une liste de tirets correspondant au mot à chercher
    devine=genere_tirets(len(motcache))
    # Trouve : variable de la boucle permettant de sortir
    Trouve = False
    while (Trouve == False) and (nberreur < ERREURMAX): # boucle par proposition de
lettre
        #saisir une lettre
        lettre=list(input("Faire une proposition de lettre :"))
        lettresProp.append(lettre[0])
        if(lettre_presente(motcache,lettre)==True):
            devoiler_lettre(motcache,devine,lettre)
            print("Trouvé",lettresProp, "   ",''.join(devine)) # join permet
d'afficher la liste devine sous forme de mot
            Trouve =test_gagner(motcache,devine)
        else:
            nberreur=nberreur+1
            print("Pas Trouvée",lettresProp, "   ",''.join(devine))
    if Trouve == True:
        print("Bravo c'est gagné en ",nberreur," érreurs")
    else:
        print("Perdu !!!  ",nberreur," érreurs")
    print("La solution était :",motcache)

if __name__ == '__main__':
    """ Fonction spéciale débutant tous les programmes
    """
    prog_principal()
```

11. Code fourni élève 1 :

```
# dictionnaire des fréquences des lettres dans un texte en français
frequence = {'a' : 0.083944, 'b' : 0.007669,  'c' : 0.033297, 'd' : 0.040699, 'e' :
0.145037,
             'f' : 0.012109, 'g' : 0.009495, 'h' : 0.007973, 'i' : 0.081828, 'j' :
0.006377, 'k' : 0.000638, 'l' : 0.058405,
             'm' : 0.029355, 'n' : 0.075570, 'o' : 0.053669, 'p' : 0.032087, 'q' :
0.012613, 'r' : 0.070209, 's' : 0.080091,
             't' : 0.074775, 'u' : 0.059808, 'v' : 0.015791, 'w' : 0.000067, 'x' :
0.004098, 'y' : 0.003155, 'z' : 0.001240}

#Candidat 1
def lire_fichier(nomFichier):
    """ ouvre et lit le fichier contenant tous les mots francais
        :param nomFichier: le nom du fichier à lire
        :type nomFichier:  String
    """
    fichier=open(nomFichier)
    texte=fichier.readlines()  #lit toutes les lignes du fichier texte
    fichier.close()
    print(texte)
```

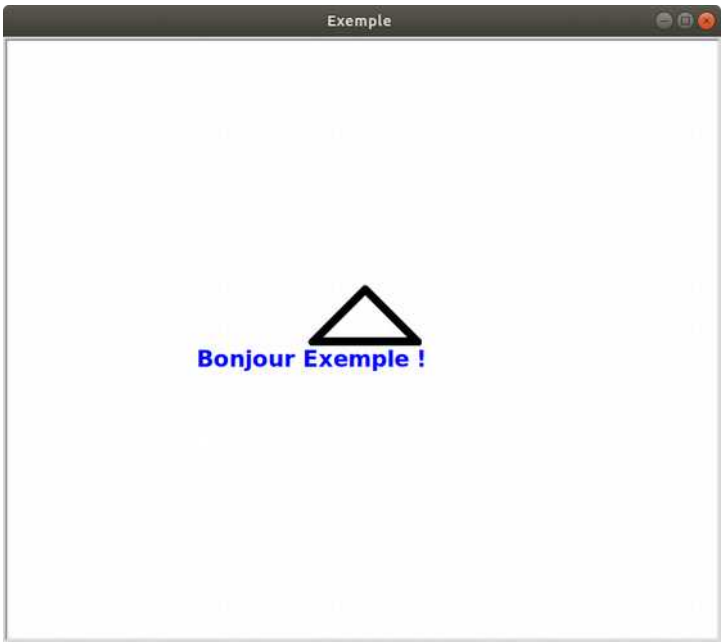
```
if __name__ == '__main__':
    """ Fonction spéciale débutant tous les programmes
    """
    print(frequence)
    lire_fichier("liste_francaisU.txt")
```

12. Code fourni élève 3 :

```
# dictionnaire des fréquences des lettres dans un texte en français
frequence = {'a' : 0.083944, 'b' : 0.007669,  'c' : 0.033297, 'd' : 0.040699, 'e' :
0.145037,
             'f' : 0.012109, 'g' : 0.009495, 'h' : 0.007973, 'i' : 0.081828, 'j' :
0.006377, 'k' : 0.000638, 'l' : 0.058405,
             'm' : 0.029355, 'n' : 0.075570, 'o' : 0.053669, 'p' : 0.032087, 'q' :
0.012613, 'r' : 0.070209, 's' : 0.080091,
             't' : 0.074775, 'u' : 0.059808, 'v' : 0.015791, 'w' : 0.000067, 'x' :
0.004098, 'y' : 0.003155, 'z' : 0.001240}

# candidat 3
def gen_html(nomFichier):
    """ créé et écrit du texte dans un fichier dont le nom est donné en paramètre
        Le texte écrit est le squelette d'un fichier html
        :param nomFichier: le nom du fichier à créer
        :type nomFichier:  String
    """
    fichier= open(nomFichier,"w")
    # écriture dans le fichier avec la méthode print
    print("""<!DOCTYPE html> <!DOCTYPE html><html lang=\"fr\"><head>  <meta
charset="UTF-8">
          <meta name="author" content="test"> <title>Test </title>
          <link rel="stylesheet" type="text/css"
href="mystyle.css"></head><body>""",end='\n',file=fichier )
    # écriture dans le fichier avec la méthode write()
    fichier.write('<h1>Bonjour à tous !</h1>')
    print('</body></html>',end='\n',file=fichier )
    fichier.close()

if __name__ == '__main__':
    """ Fonction spéciale débutant tous les programmes
    """
    gen_html("pageTest.html")
```

13. Code fourni élève 2 :

```
iimport turtle

def prog_principal(ihm):
    """ permet de tracer 3 segments du triangle, d'afficher un texte, de demander une
saisie,
        d'éffacer le premier texte et de le remplacer par le texte saisi
        L'ensemble des fonctions de turtle utilent pour le pendu sont présentes.
        :param ihm: l'objet permettant de faire des changements sur l'écran
        :type ihm: TurtleScreen
    """

    stylo1= turtle.Turtle()
    stylo1.hideturtle() # cache le pointeur permettant de suivre la position de
tracage
    stylo1.penup()
    stylo1.pensize(8)
    stylo2= turtle.Turtle()
    stylo2.hideturtle()# cache le pointeur permettant de suivre la position de
tracage

    debut = (-50,0)
    fin = (50,0)
    stylo1.penup()
    stylo1.goto(debut)
    stylo1.pendown()
    stylo1.goto(fin)
    trait=[(0,50),(50,0)]
    stylo1.penup()
    stylo1.goto(trait[0])
    stylo1.pendown()
    stylo1.goto(trait[1])
    trait2=[(-50,0),(0,50)]
    stylo1.penup()
    stylo1.goto(trait2[0])
    stylo1.pendown()
    stylo1.goto(trait2[1])

    stylo2.penup()
    stylo2.goto(-50,-30)
    couleur='blue'
    stylo2.color(couleur)
    msg="Bonjour Exemple !"
    stylo2.write(msg, align="center", font=(None, 16, "bold"))
    msg2= ihm.textinput("Réponse!"," Réponse: ")
    couleur='red'
    stylo2.color(couleur)
    stylo2.clear() # efface les précédente action de stylo2
    stylo2.write(msg2, align="center", font=(None, 16, "bold"))


def init_fenetre():
    """ permet d'initialiser la fenêtre avec une taille sa position sur l'écran son
titre
        :return: ihm l'objet de type screen permettant de faire des actions sur la
fenêtre
        :rtype: TurtleScreen
    """
    ihm =turtle.Screen()
```



```
    ihm.setup(640, 480, 100, 100)  #Largeur : 640px, Hauteur : 480px, pos x : 100px,
pos y : 100px
    ihm.setup(200, 200)  #Largeur : 200px, Hauteur : 200px, position centrée
    ihm.setup(startx = 0, starty = 0)  #Largeur : 50%, Hauteur : 75%, position : coin
haut gauche écran
    ihm.setup()  #Largeur : 50%, Hauteur : 75%, position centrée
    ihm.title("Exemple ")  #Change le titre
    return ihm


if __name__ == '__main__':
    """ Fonction spéciale débutant tous les programmes
    """
    ihm=init_fenetre()
    prog_principal(ihm)

    ihm.exitonclick()
    turtle.bye()
```

14. Listing presque complet de la solution du projet :

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jun 14 09:22:08 2019

@author: magali.sauvergeat
"""
from random import randint
from random import choice
import turtle
import os.path

ERREURMAX =12
# partie 3
def gen_html(frequence,alpha,nomfichier):
  """ créé et génére un fichier html présentant le tableau des fréquences
      de l'alphabet donné en entrée
        :param frequence: dictionnaire associant une lettre avec sa fréquence
        :type frequence: dictionnaire
        :param alpha: la liste des lettres de l'alphabet
        :type alpha: liste
        :param nomfichier: le nom du fichier à créer
        :type nomfichier:  String
  """
  try:
        #nomfichier="pageFrequence.html"
        print(nomfichier)
        fichier= open(nomfichier,"w")
        print("""<!DOCTYPE html> <!DOCTYPE html><html lang=\"fr\"><head>  <meta
charset="UTF-8">  <meta name="author" content="msauver">
        <title>Table des fréquences de l'alphabet en français </title> <link
rel="stylesheet" type="text/css" href="mystyle.css">
        </head><body><h1> Table des fréquences de l'alphabet en français </h1>
        <section><table><tr class="lumiere"><th>Lettre</th> <th> % de Fréquence</th>
<th>Lettre</th> <th>% de Fréquence</th></tr>""",end='\n',file=fichier )
        for i in range(len(alpha)//2):
            print('<tr><td class="lumiere">',alpha[i],end='\n',file=fichier )
            print('</td><td> %.2f</td><td class="lumiere">'%
(frequence[alpha[i]]*100),end='\n',file=fichier )
            print(alpha[i+(len(alpha)//2)],end='\n',file=fichier )
```

```
            print('</td><td>%.2f</td></tr>'
%(frequence[alpha[i+(len(alpha)//2)]]*100),end='\n',file=fichier )

        print('</table></section>',end='\n',file=fichier )
        print('<footer>DIU-Bloc1  </footer></body></html>',end='\n',file=fichier )
        fichier.close()
  except FileNotFoundError:
        print("Erreur ouverture de Fichier")

def gen_htmlP(frequence,alpha,nomfichier):
  """ créé et génére un fichier html présentant le tableau des probabilité
d'apparition
      d'une lettre dans un mot
        :param frequence: dictionnaire associant une lettre avec sa probabilité
d'apparition dans un mot
        :type frequence: dictionnaire
        :param alpha: la liste des lettres de l'alphabet
        :type alpha: liste
        :param nomfichier: le nom du fichier à créer
        :type nomfichier:  String
  """
  try:
        #nomfichier="pageFrequence.html"
        print(nomfichier)
        fichier= open(nomfichier,"w")
        print("""<!DOCTYPE html> <!DOCTYPE html><html lang=\"fr\"><head>  <meta
charset="UTF-8">  <meta name="author" content="msauver">
        <title>Table des probabilités d'apparition d'une lettre dans un mot</title>
<link rel="stylesheet" type="text/css" href="mystyle.css">
        </head><body><h1> Probabilité d'apparition d'une lettre dans un mot
</h1>""",end='\n',file=fichier )
      #  print("<h1> dans le fichier de test </h1>",end='\n',file=fichier )
        print("""<section><table><tr class="lumiere"><th>Lettre</th> <th>% de
Probabilité</th> <th>Lettre</th> <th>% de Probabilité</th></tr>""",end='\
n',file=fichier )
        for i in range(len(alpha)//2):
            print('<tr><td class="lumiere">',alpha[i],end='\n',file=fichier )
            print('</td><td> %.2f</td><td class="lumiere">'%
(frequence[alpha[i]]*100),end='\n',file=fichier )
            print(alpha[i+(len(alpha)//2)],end='\n',file=fichier )
            print('</td><td>%.2f</td></tr>'
%(frequence[alpha[i+(len(alpha)//2)]]*100),end='\n',file=fichier )

        print('</table></section>',end='\n',file=fichier )
        print('<footer>DIU-Bloc1  </footer></body></html>',end='\n',file=fichier )
        fichier.close()
  except FileNotFoundError:
        print("Erreur ouverture de Fichier")


def gen_dico_vide(freqalpha,alpha):
    """générer un dictionnaire vide associant 0 avec les lettres présentes dans
        la liste alpha
        :param freqalpha: dictionnaire vide en entrée modifié dans la fonction
                          associant une lettre de alpha avec la valeur 0
        :type freqalpha: dictionnaire
        :param alpha: la liste des lettres de l'alphabet
        :type alpha: liste
    """
    for i in range(len(alpha)):
         freqalpha[alpha[i]] = 0
```

```
    print(freqalpha)


def calcul_frequence(freqalpha, nomfichier):
    """ compléter le dictionnaire vide de fréquence d'une lettre en utilisant
        un fichier de mots pour la fréquence
        :param nomFichier: le nom du fichier à lire
        :type nomFichier:  String
        :param freqalpha: dictionnaire vide en entrée modifié dans la fonction
                          associant une lettre avec sa fréquence dans le fichier
        :type freqalpha: dictionnaire
    """
    #nomfichier="liste_francaisU.txt"
    if os.path.isfile(nomfichier): # retourne vrai si le fichier existe
        print(nomfichier)
        fichier= open(nomfichier,"r")
        mot ="debut"
        nbl = 0
        while mot !="" :
            mot = fichier.readline()
            mot=normalise(mot)
            for lettre in mot:
                if lettre in freqalpha.keys():
                    freqalpha[lettre]=freqalpha[lettre]+1
                    nbl= nbl+1
        print(freqalpha)
        print("nombre de lettres",nbl)
        fichier.close()
        if nbl>0:
            for l in freqalpha.keys():
                freqalpha[l] = freqalpha[l]/nbl
            print(freqalpha)

def calcul_proba_lettre(probaalpha,nomfichier):
    """ compléter le dictionnaire vide de probabilité d'une lettre en utilisant
        un fichier de mots pour la probabilité
        :param nomFichier: le nom du fichier à lire
        :type nomFichier:  String
        :param probaalpha: dictionnaire vide en entrée modifié dans la fonction
                          associant une lettre avec sa probabilité d'apparition
dans un mot
        :type probaalpha: dictionnaire
    """
    #nomfichier="liste_francaisU.txt"
    if os.path.isfile(nomfichier): # retourne vrai si le fichier existe
        print(nomfichier)
        fichier= open(nomfichier,"r")
        mot ="init"
        nbm = -1 # compteur de mot
        while mot !="" :
            mot = fichier.readline()
            mot=normalise(mot)
            unique=set(mot) # enlève les lettres redondantes
            nbm= nbm+1
            for lettre in unique:
                if lettre in probaalpha.keys():
                    probaalpha[lettre] = probaalpha[lettre]+1

        print(probaalpha)
        print("nombre de mots",nbm)
        fichier.close()
```

```
        if nbm>0:
            for l in probaalpha.keys():
                probaalpha[l] = probaalpha[l]/nbm
            print(probaalpha)

def demo_v2(lettres, frequence ):
    """ Trouver la lettre dont la fréquence est maximum, qui n'a pas déjà été
proposée.
        :param lettres: la liste des lettres déjà proposées
        :type lettres: liste
        :param frequence: dictionnaire associant une lettre avec sa fréquence
        :type frequence: dictionnaire
        :return l:  la lettre
        :rtype: string
    """
    maxF=0.0
    maxL=""
    for clef, valeur in frequence.items():
        #test clef n'est pas dans lettres
        Trouve=False
        for i in range(len(lettres)):
            if clef == lettres[i]:
                Trouve=True
        if (maxF < valeur) and (Trouve == False):
            maxF=valeur
            maxL=clef
    return maxL

def demo_v1(lettres):
    """ tire une lettre au hasard vérifie qu'elle n'a pas été proposée
        retourne la lettre
        :param lettres: la liste des lettres déjà proposées
        :type lettres: liste
        :return l:  la lettre
        :rtype: string
    """
    l=""
    Trouve=True
    alpha =
['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']
    while Trouve==True:
        Trouve=False
        l=choice(alpha) #tirer aléatoirement une lettre
        for i in range(len(lettres)):
            if l == lettres[i]:
                Trouve=True
    return l

def choisi_mot(nbmax,nomfichier):
    """ ouvre et lit le fichier contenant des mots francais
        tire un mot au hasard vérifie qu'il ne dépasse pas nbmax en longueur
        élimine les accents et retourne le mot
        :param nbmax: le nombre de lettre maximum du mot
        :type nbmax:  entier integer
        :param nomfichier: le nom du fichier à lire
        :type nomFichier:  String
        :return: inconnu le mot à deviner ou vide si fichier n'existe pas
        :rtype: string
    """
    if os.path.isfile(nomfichier): # retourne vrai si le fichier existe
```

```
        fichier=open(nomfichier)
        texte=fichier.readlines()  #lit toutes les lignes du fichier texte
        fichier.close()
        taille=len(texte)

        longueur=-1
        #tirage au sort d'un mot inconnu entre 6 et 3 lettres
        while (longueur>nbmax) or (longueur<3):
            nombre = randint(0,taille-1)
            inconnu=texte[nombre]
            inconnu=normalise(inconnu) # enlève les accents
            inconnu=inconnu.strip() #retire le \n final
            longueur=len(inconnu)
        return inconnu
    else:
        return "vide"

def normalise(inconnu):
    """ prend un mot avec accent et autres lettres spéciales
        retourne le mot sans accent
        :param inconnu: le mot avec accent
        :type inconnu:  string
        :return: inconnu le mot sans accent
        :rtype: string
    """
    inconnu=inconnu.lower()
    inconnu=inconnu.replace('é','e')
    inconnu=inconnu.replace('ê','e')
    inconnu=inconnu.replace('î','i')
    inconnu=inconnu.replace('ï','i')
    inconnu=inconnu.replace('è','e')
    inconnu=inconnu.replace('û','u')
    inconnu=inconnu.replace('à','a')
    inconnu=inconnu.replace('ç','c')
    inconnu=inconnu.replace('ô','o')
    # réfléchir au carctère e dans l'o
    return inconnu

def genere_tirets(n):
    """ permet de générer une liste de tirets représentant les lettres à trouver
        :return: ch
        :rtype: liste
    """
    ch=[]
    for i in range(n):
        ch.append("-")
    return ch

def lettre_presente(tab,l):
    """ vérifier si la lettre proposée est dans le mot à trouver
        :param tab: la liste des lettres du mot à trouver
        :type tab:  list
        :param l: la proposition dans la première position d'une liste
        :type l: list
        :rtype: booleen
    """
    for i in range(len(tab)):
        if(tab[i]==l[0]):
            return True
    return False
```

```
def devoiler_lettre(sol,prop,l):
    """ Montre la lettre dans le tableau de proposition
        :param sol: la liste des lettres du mot à trouver
        :type sol:  list
        :param prop: cette liste sera modifiée en remplaçant les tirets par la lettre
        :type prop:  list
        :param l: la proposition dans la première position d'une liste
        :type l: list
    """
    for i in range(len(sol)):
        if(sol[i]==l[0]):
            prop[i]=sol[i]

def test_gagner(sol,prop): # if(sol == prop)
    """ Vérifie si la proposition est bonne
        :param sol: la liste des lettres du mot à trouver
        :type sol:  list
        :param prop: cette liste sera modifiée en remplaçant les tirets par la lettre
        :type prop:  list
        :rtype: booleen
    """
    for i in range(len(sol)):
        if(sol[i]!=prop[i]):
            return False
    return True

def dessin_pendu(feutre,etape, placement):
    """ dessine la partie du pendu correspondant à l'étape
        avec l'objet feutre permettant de tracer le pendu.
        :param feutre: l'objet permettant de dessiner sur la fenêtre
        :type feutre:  objet turtle
        :param etape: le numéro de l'étape correspondant aux nombres d'érreurs
        :type etape:  entier integer
        :param placement: liste des coordonnées des traits(segements) du pendu
        :type placement:  list
    """
    if etape < 12:
        feutre.penup()
        if etape != 6 :
            feutre.goto(placement[etape*2])
            feutre.pendown()
            feutre.goto(placement[etape*2+1])
        else:               # cas de la tête
            feutre.goto(placement[etape*2])
            feutre.pendown()
            feutre.dot(30)
            feutre.pensize(5)  # pour le corps on passe à un trait plus fin



def dessin_texte(stylo,texte,position,couleur="black",efface=True):
    """ dessine du texte sur l'écran à une position, avec une couleur
        efface l'ancien message par defaut
        :param stylo: l'objet permettant de dessiner sur la fenêtre
        :type stylo:  objet turtle
        :param texte: le message à écrire sur la fenêtre
        :type texte:  string
        :param position: une coordonnées de centrage du texte
        :type position: couple d'entiers (x,y)
        :param couleur: mot représentant une couleur par defaut "black"
        :type couleur:  string
```

```
        :param efface: précise si l'ancien message doit être éffacé par defaut True
        :type efface:  booleen
    """
    stylo.goto(position)
    if efface:
        stylo.clear()
    stylo.color(couleur)
    stylo.write(''.join(texte), align="center", font=(None, 16, "bold"))

########################## PROG PRINCIPAL EN BOUCLE
#####################################################"

def prog_principal(ihm,probaalpha):
    #liste des coordonées des segments du pendu
    placement = [(0,-70), (0,-100), # placement du texte
            (-150,-30),(50,-30),#plancher
             (-50,-30),(-50,150),#barre verticale
             (-50,150),(80,150),#barre horirontale haute
             (-50,100),(30,150),#barre oblique
             (80,150),(80,120),#haut corde
             (80,110),(80,110),#tete
            (80,100),(80,30), #colonne vertébrale
            (80,30),(60,10),#jambe droite
            (80,30),(100,10),#jambe gauche
            (80,90),(60,70),  #bras droit
            (80,90),(100,70) ]  #bras gauche

    mode=0    # variable permettant de dire si mode utilisateur 0 ou mode démo 1 et 2
    feutre= turtle.Turtle() # pour le dessin du pendu
    feutre.hideturtle() # cache le pointeur permettant de suivre la position de
tracage
    stylo= turtle.Turtle() # pour afficher le tableaux de tirets
    stylo.hideturtle()# cache le pointeur permettant de suivre la position de tracage
    stylo2= turtle.Turtle() # pour afficher les lettres déjà proposées
    stylo2.hideturtle()# cache le pointeur permettant de suivre la position de
tracage
    while mode != 3 :       # boucle pour rejouer
        # fenetre pour le choix du mode
        msg="Choisir Mode :\n 0 - utilisateur \n 1 - demo_v1\n 2 - demo_v2 \n 3 -
Quitter "
        mode = int(ihm.numinput("Mode de Jeu ", msg,minval = 0, maxval = 3))
        if mode == 3:
            return
        # liste contenant les lettres déjà propsées
        lettresP=[]
        # compteur du nombre d'érreurs
        nberreur=0
        #partie 1
        motcache= choisi_mot(7,"liste_francaisU.txt")
        feutre.penup()
        feutre.pensize(8)
        #pour le rebouclage
        feutre.clear()
        stylo.clear()
        stylo2.clear()
        # Titre
        msg ="Le Pendu DIU-Bloc1 Orsay"
        dessin_texte(feutre,msg,(-50,170),'blue',False)
        feutre.color("black")
        # génère une liste de tirets correspondant au mot à chercher
        devine=genere_tirets(len(motcache))
```

```
        #partie 2
        dessin_texte(stylo,devine,placement[0])
        Trouve = False
        while (Trouve == False) and (nberreur <ERREURMAX):  # boucle par proposition
de lettre
            if mode == 0 :
                saisieCorrecte = False
                while saisieCorrecte != True :
                    msg="Lettres déjà proposées"+str(lettresP)+"\nProposition de
lettre: "
                    lettre = ihm.textinput("Devinez !!!", msg)
                    if lettre != None and len(lettre) >0 and lettre[0]>='a' and
lettre [0]<='z':
                        lettre = list(lettre)
                        saisieCorrecte = True
            #partie 1
            elif mode == 1 :
                lettre = demo_v1(lettresP)
            elif mode == 2 :
                lettre = demo_v2(lettresP,probaalpha)
            lettresP.append(lettre[0])
            if(lettre_presente(motcache,lettre)==True):
                devoiler_lettre(motcache,devine,lettre)
                dessin_texte(stylo,devine,placement[0])
                msg = "lettres proposées :"+''.join(lettresP)
                dessin_texte(stylo2,msg,placement[1])
                #print("Trouvé",lettresP, "   ",''.join(devine))
                Trouve =test_gagner(motcache,devine)
            else:
                nberreur=nberreur+1
                dessin_pendu(feutre,nberreur,placement) # dessine la partie du pendu
correspondant
                dessin_texte(stylo,devine,placement[0])
                msg = "lettres proposées :"+''.join(lettresP)
                dessin_texte(stylo2,msg,placement[1])
                #print("Pas Trouvée",lettresP, "   ",''.join(devine))
        if Trouve == True:
                msg = "Gagné ! \n La solution était : "+motcache
                dessin_texte(stylo2,msg,placement[1],'red')
        else:
                msg = "Perdu ! \n La solution était : "+motcache
                dessin_texte(stylo2,msg,placement[1],'red')

def init_fenetre():
    """ permet d'initialiser la fenêtre avec une taille sa position sur l'écran son
titre
        :return: ihm l'objet de type screen permettant de faire des actions sur la
fenêtre
        :rtype: TurtleScreen
    """
    ihm =turtle.Screen()
    ihm.setup(640, 480, 100, 100)  #Largeur : 640px, Hauteur : 480px, pos x : 100px,
pos y : 100px
    ihm.setup(200, 200)  #Largeur : 200px, Hauteur : 200px, position centrée
    ihm.setup(startx = 0, starty = 0)  #Largeur : 50%, Hauteur : 75%, position : coin
haut gauche écran
    ihm.setup()  #Largeur : 50%, Hauteur : 75%, position centrée
    ihm.title("Le Pendu ")  #Change le titre
    return ihm
```

```
########################## LANCEMENT DU PROG ET INITIALISATION
#####################################################"
if __name__ == '__main__':
    """ Fonction spéciale débutant tous les programmes
    """
    # dictionnaire des fréquences des lettres dans un texte en français
    frequence = {'a' : 0.083944, 'b' : 0.007669,  'c' : 0.033297, 'd' : 0.040699, 'e'
: 0.145037,
              'f' : 0.012109, 'g' : 0.009495, 'h' : 0.007973, 'i' : 0.081828, 'j' :
0.006377, 'k' : 0.000638, 'l' : 0.058405,
              'm' : 0.029355, 'n' : 0.075570, 'o' : 0.053669, 'p' : 0.032087, 'q' :
0.012613, 'r' : 0.070209, 's' : 0.080091,
              't' : 0.074775, 'u' : 0.059808, 'v' : 0.015791, 'w' : 0.000067, 'x' :
0.004098, 'y' : 0.003155, 'z' : 0.001240}

    freqalpha = {}
    probaalpha = {}
    alpha =
['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']


    #partie 3
    gen_dico_vide(freqalpha,alpha)
    calcul_frequence(freqalpha,"liste_francaisU.txt")
    gen_dico_vide(probaalpha,alpha)
    calcul_proba_lettre(probaalpha,"liste_francaisU.txt")
    gen_htmlP(probaalpha,alpha,"pageFrequenceProba.html")
    gen_html(freqalpha,alpha,"pageFrequenceCalcul.html")
    gen_html(frequence,alpha,"pageFrequence.html")

#    #partie 2
    ihm=init_fenetre()
    prog_principal(ihm,probaalpha)

    ihm.exitonclick()
    turtle.bye()
```