6 - Interfaces externes

1. Utilisation de fichiers

2. Interfaces graphiques

6.1 - Utilisation de fichiers

- Il est important de dissocier les données des programmes qui les utilisent en rangeant ces données dans des fichiers séparés.

- Le module **os** contient des fonctions qui permettent de localiser les fichiers :

getcwd()	Retourne le chemin du répertoire courant
chdir(<*ch*>)	Change le répertoire courant qui prend la valeur donnée par la chaîne de caractères <*ch*>
path.isfile(<*ch*>)	Retourne un booléen qui indique s'il existe un fichier dont le chemin est la chaîne de caractères <*ch*>
path.isdir(<*ch*>)	Retourne un booléen qui indique s'il existe un répertoire dont le chemin est la chaîne de caractères <*ch*>

6.1 - Utilisation de fichiers

- Pour utiliser un fichier identifié par le chemin *ch* dans un programme Python, il faut commencer par l'ouvrir par l'appel de fonction *open(<ch>, [<mode>])* qui retourne un objet de type *file*. Le paramètre facultatif *<mode>* indique le mode d'ouverture du fichier :

  * 'r' : mode lecture (le fichier doit exister préalablement)

  * 'w' : mode écriture (si le fichier existe, les données sont écrasées, sinon le fichier est créé)

  * 'a' : mode ajout (si le fichier existe, les données écrites vont l'être après celles existantes, sinon le fichier est créé)

  Si le mode est omis, le mode par défaut est 'r'.

6.1 - Utilisation de fichiers

- Comme tout objet Python, un objet de type file est associé à des attributs et des méthodes. En voici quelques-unes :

read(*[<n>]*)	Retourne la chaîne des <n> caractères restants.
write(<*ch*>)	Ecrit la chaîne de caractères <*ch*>.
close()	Ferme le fichier quand il est fini d'être utilisé.
seek(<*n*>)	Choisit le caractère <*n*> comme position courante du fichier.

6.1 - Utilisation de fichiers

```
>>> from os import chdir
>>> chdir('Users/perrier/Desktop')
>>> getcwd()
'Users/perrier/Desktop'
>>> from os import path
>>> path.isfile('./test')
True
>>> f = open('./test', 'r')
>>> f.read(3)
'bon'
>>> f.read()
'jour'
>>> f.seek(0)
>>> f.read()
'bonjour'
>>> f.close()
```

```
>>> f2 = open('./test', 'a')
>>> f2.write(' cher ami')
>>> f2.close()
>>> f2 = open('test', 'r')
>>> f2.read()
'bonjour cher ami'
>>> f2.close()
>>>
```

6.2 - Interfaces graphiques

- Python dispose d'un module **Tkinter** qui est une interface avec **Tk** de **Tcl** pour créer et gérer des interfaces graphiques

- Documentation : *http://www.pythonware.com/library/tkinter/introduction/*.

- Le module **Tkinter** fournit la classe d'objets **Tk** constituée de fenêtres.

- Il fournit aussi des **contrôles (widgets)** qui peuvent être placés dans ces fenêtres. Tkinter offre 15 classes de base de contrôle. Un contrôle est un objet graphique qui permet une interaction sous une forme spécifique avec l'utilisateur : ce dernier, à l'aide du clavier ou de la souris crée des évènements déclencheurs de programmes Python. On parle alors de **programmation évènementielle**.

6.2 - Interfaces graphiques

- On peut intégrer un contrôle stocké dans la variable *x* dans un contrôle ou une fenêtre maître *y* en passant le second en argument à la création du premier, à l'aide de l'instruction : *x = <classe de contrôle>(y, …)*

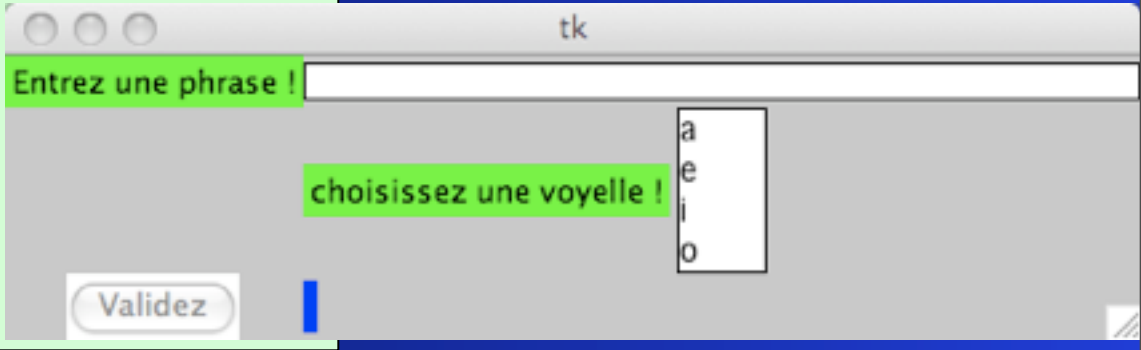
- Le positionnement d'un contrôle dans sa fenêtre d'affichage se fait soit à l'aide de la méthode **pack**, soit de façon plus précise à l'aide de la méthode **grid** qui est fondée sur le découpage de la zone de placement en une grille à deux dimensions.

- Le liage d'un contrôle stocké dans la variable *x* avec un événement *e* et une procédure évènementielle *f* s'effectue à l'aide de la méthode **bind** selon la syntaxe suivante : *x.bind(e,f).* La procédure *f* est définie comme une fonction Python avec comme argument l'événement *e* et ne retournant aucune valeur.

6.2 - Interfaces graphiques

```
from Tkinter import *

# définition des contrôles et de leurs relations
fen1 = Tk()
fr1 = Frame(fen1, bg='grey')
list1= Listbox(fr1, width=4, height=4)
ent1 = Entry(fr1, width=50)
lab1= Label(fr1, text= "Entrez une phrase !", bg ='green')
lab3= Label(fr1, text="choisissez une voyelle !", bg='green')
v=StringVar()
lab4 = Label(fr1, textvariable=v, bg='blue')
list1.insert(END, 'a')
list1.insert(END, 'e')
list1.insert(END, 'i')
list1.insert(END, 'o')
list1.insert(END, 'u')
but1= Button(fr1, text="Validez")
```



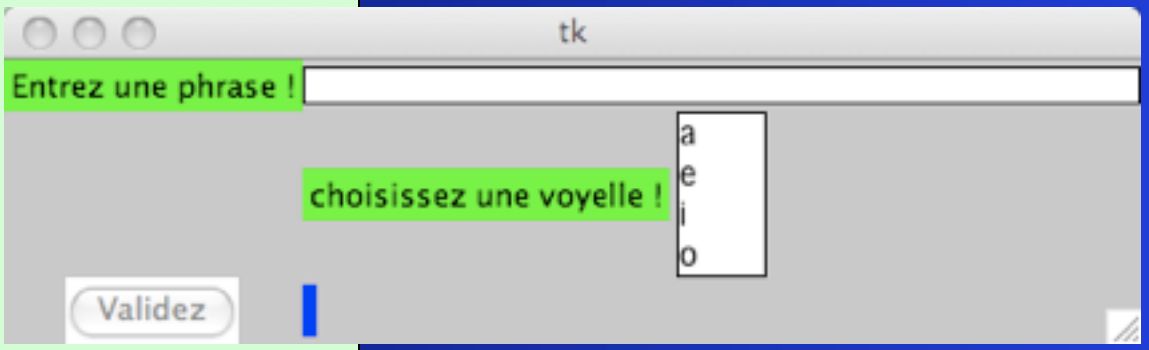
6.2 - Interfaces graphiques

```
# positionnement des contrôles
fr1.pack()
lab1.grid(row=0, column=0)
ent1.grid(row=0, column=1)
list1.grid(row=1, column=1)
lab3.grid(row=1, column=1)
but1.grid(row=2, column=0)
lab4.grid(row=2, column=1)

#définition des procédures événementielles
def compter(event):
    texte = ent1.get()
    lettre_selectionnee = list1.selection_get()
    n=0
    for c in texte:
        if c == lettre_selectionnee:
            n +=1
    v.set('nombre de voyelles ' + lettre_selectionnee + ' : ' +str(n))

# liage des procédures aux contrôles et événements
but1.bind('<Button-1>', compter)

#activation de l'interface graphique
fen1.mainloop()
```



6.3 - Exercices

1. Ecrire des programmes Python qui réalisent les spécifications suivantes.

a) A partir d'un texte stocké dans un fichier identifié par le chemin *./corpus/texte1.txt* créer un dictionnaire ordonné des mots fléchis qu'il contient et stocker ce dictionnaire dans le fichier *./lexiques/lexique1.txt* (mettre un mot par ligne).
Aide : si on utilise un dictionnaire, la méthode keys() *fournit la liste les clés*

b) On veut définir une classe *Lexiques* dont les instances sont des lexiques morphologiques. Ces lexiques sont rangés dans des fichiers avec une entrée par ligne. Une entrée se présente selon la syntaxe suivante :
<mot> : <lemme>, <catégorie>, {param$_1$: val$_1$, param$_2$: val$_2$,…param$_n$: val$_n$}
Les entrées sont ordonnées selon l'ordre lexicographique des mots.
Créer une classe *Lexiques* avec comme attributs : *nom, emplacement.* Créer aussi la méthode *extraire_entree(m)* qui extrait du lexique l'entrée du mot *m*, et la méthode *inserer_entree(entr)* qui insère dans le lexique l'entrée *entr* supposée dans un format correct.

6.3 - Exercices

2. Ecrire des programmes Python qui réalisent les spécifications suivantes à l'aide d'interfaces graphiques.

a) On saisit le chemin d'un fichier stockant un lexique morphologique dans une zone de texte. La frappe de la touche Entree est un événèment <*Return*> qui provoque l'ouverture du fichier choisi. Puis on saisit un mot dans une zone de texte et on choisit sa catégorie grammaticale dans une liste de choix. La sélection de la catégorie par un double click gauche est un événement <*Double-Button-1*> qui entraîne l'affichage de l'entrée correspondante du lexique ouvert (le format du lexique est celui défini dans l'exercice précédent).

b) On réalise un traducteur de mots entre le français et l'anglais dans les deux sens à l'aide de deux zones de texte et de deux boutons de validation correspondant aux deux sens de traduction. On dispose pour cela d'un seul dictionnaire franco-anglais rangé dans un fichier selon l'ordre lexicographique avec une entrée par ligne.