6 - External interfaces

1. Manipulation of files

2. Graphic interfaces

6.1 - Manipulation of files

- It is important to dissociate data from programs that use them by storing data in files independent of programs.

- The **os** module contains functions that are used for locating files:

getcwd()	Returns the path of the current directory.
chdir(<*path*>)	Changes the current directory with path <*path*>
path.isfile(<*path*>)	Returns a boolean which indicates if the file located at path <*path*> exists or not.
path.isdir(<*path*>)	Returns a boolean which indicates if the directory located at path <*path*> exists or not.

6.1 - Manipulation of files

- A file identified by means of the path <*path*>, to be used in a Python program, must be opened through the function call *open(<path>, [<mode>]),* which returns a object of type *file*. The optional parameter <*mode*> indicates the opening mode of the file:

  * ‘r’ : read mode (the file must exist first);
  * ‘w’ : write mode (if the file exists, its data are overwritten, otherwise the file is created);
  * ‘a’ : addition mode (if the file exists, new data will be written after stored data, otherwise the file is created).

  If the mode is omitted, the default mode is ‘r’.

6.1 - Manipulation of files

- As every Python object, an object of type file is associated with attributes and methods. Here are some methods associated with files:

read(*[<n>]*)	Returns the string of the last <n> characters from the file.
write(<*s*>)	Writes string <*s*>.
close()	Closes the file.
seek(<*n*>)	Put the pointer of the file at position <*n*>.

6.1 - Manipulation of files

```
>>> 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 - Graphic interfaces

- Python has a module, **Tkinter,** which is an interface between Python and **Tk** of **Tcl** used to create and to manage graphic interfaces.
- Documentation : *http://www.pythonware.com/library/tkinter/introduction/*.
- The **Tkinter** module provides the **Tk** object class, the instances of which are graphic windows.
- It also provides **widgets** which can be placed in these windows. Tkinter offers 15 classes of widgets. A widget is a graphic object that enables a user to interact with a program in a specific form: with the keyboard or the mouse, the user can create events triggering Python programs. This mechanism is called **event programming**.

6.2 - Graphic interfaces

- A widget stored in a variable x can be integrated in another widget or a window y in the following way: y is passed as argument in the instruction of the widget creation

  x = <widget class>(y, …)

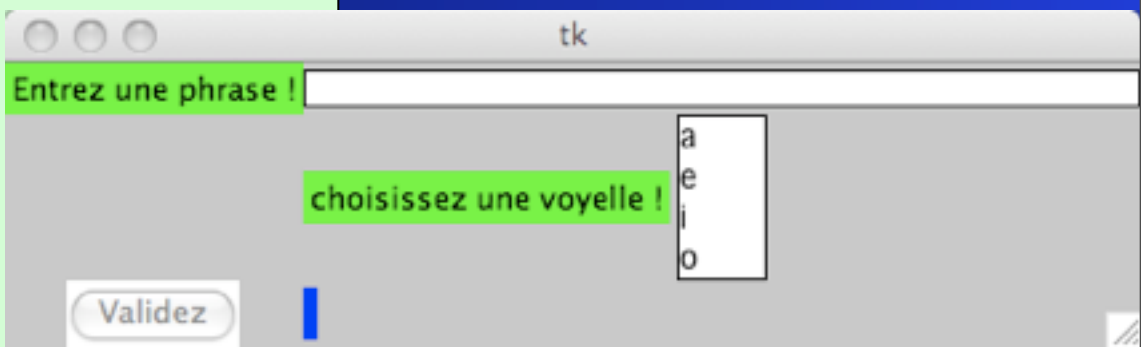
- The positioning of a widget in its window is performed by means of the **pack** method or more precisely with the **grid** method, which is based on the division of the window according to a two dimension grid.

- The linking of a widget stored in a variable x with an event e and an event procedure f is performed through the **bind** method according to the following syntax: *x.bind(e,f).* The procedure f is defined as any Python function with e as parameter and it returns no value.

6.2 - Graphic interfaces

```
from Tkinter import *

# definition of the widget and their 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")
```



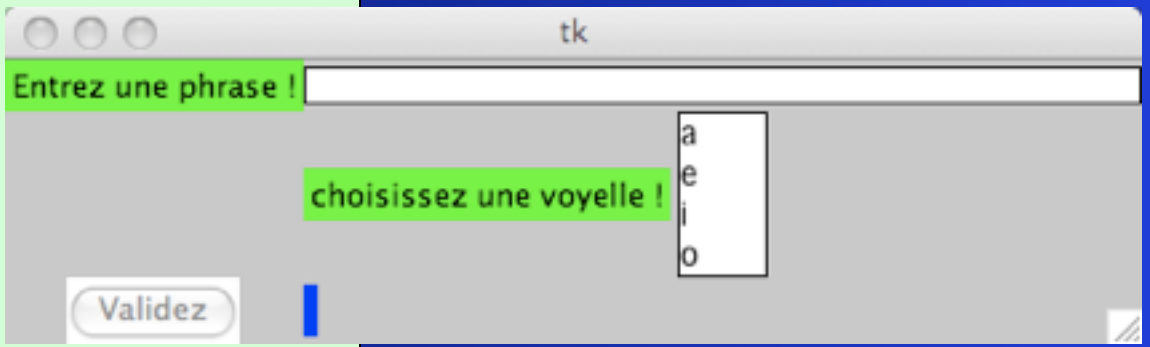
6.2 - Graphic interfaces

```
# positioning of the widgets
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)

#definition of the event procedures
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))

# linking of the event procedures with the events and widgets
but1.bind('<Button-1>', compter)

#activation of the graphic interface
fen1.mainloop()
```



6.3 - Exercises

1. Write Python programs that realize the following specifications:

   a) In a text stored in the file *./corpus/text1.txt* , replace a word entered at the keyboard with another word entered at the keyboard. The new text must be stored in the same file.

   b) From a text stored in the file *./corpus/text1.txt* , create a dictionary of its inflected words with the number of occurrences in the text for each word and store the dictionary in the file *./lexiques/lexique1.txt* (with one entry per line).

2. Write a Python program that realizes a word translator between French and English in the form of a graphic interface. In the interface, there must be two text entries and two buttons for the two directions of translation. A unique French-English dictionary is available in a file for the two directions of translation.