



```
print("Hello, world!")
```

FORMATION PYTHON : LES FICHIERS

Ce chapitre a pour but d'initier à la manipulation des fichiers en Python.

Introduction

- ▶ Les variables sont simples à utiliser, mais elles ne contiennent que des informations **temporaires**.
- ▶ La durée de vie d'une variable n'est en effet jamais très longue. Or, vous aurez certainement besoin dans vos programmes de stocker des informations définitivement.
- ▶ Par exemple, il est impossible de stocker les résultats d'un programme dans des variables... puisque celles-ci seront supprimées à la fermeture du Shell Python !
- ▶ Par exemple, un petit jeu peut enregistrer les scores des joueurs.

Introduction

- ▶ Pour stocker ces informations longtemps, il faut les écrire sur le disque dur : créer des fichiers.
- ▶ Python permet d'enregistrer des données dans des fichiers.
- ▶ Donc, il est indispensable de savoir manipuler des fichiers avec PYTHON.
- ▶ On va s'intéresser à la manipulation des fichiers texte contenant du texte (chaînes de caractères) et des fichiers images.
- ▶ Nous allons voir dans ce chapitre les fichiers : comment les ouvrir, les lire et écrire dedans.

Plan du cours

1. Introduction : Un fichier ? Types de fichiers ?
2. Fichier Texte
 1. Écrire des données dans un fichier
 2. Lecture des données depuis un fichier
3. Fichier image
 1. Écrire des données dans un fichier
 2. Lecture des données depuis un fichier

Introduction

Un fichier en informatique ! C'est quoi ?

Un fichier informatique est une collection d'informations numériques réunies sous un même nom, enregistrées sur un support de stockage permanent, appelé mémoire de masse, tel qu'un disque dur, un CD-ROM, ou une mémoire flash , et manipulées comme une unité.

Introduction

- ▶ En vue de faciliter leur organisation, les fichiers sont disposés dans des *systemes de fichiers* qui permettent de placer les **fichiers** dans des emplacements appelés **répertoires** ou **dossiers**.
- ▶ Un fichier comporte un *nom de fichier* qui sert à désigner le contenu et y accéder.
- ▶ Ce nom comporte souvent un suffixe, l'*extension*, qui renseigne sur la nature des informations contenues dans le fichier et donc des logiciels utilisables pour le manipuler.
- ▶ La nature du contenu peut être des *textes*, des *images*, de *l'audio* ou de la *vidéo*.

Introduction

7

Quelques catégories de fichiers :

Catégories de fichiers	Extensions ⁸
Exécutables (programmes exécutables)	.exe, .com, .bat
Compressés (fichier codé : Démeunier taille)	.arc, .zip, .rar, .z, .arj, .sit, .gz, .7z
Images (photos, diagrammes)	.gif, .jpg, .bmp, .png, .eps, .tif
Audio (chanson, musique)	.au, .wav, .ra, .ram
Vidéo (film)	.avi, .mpg, .mov

Introduction

Quelques catégories de fichiers :

Catégories de fichiers	Extensions ⁸
Documents : destinés à être imprimés et lus. Texte + informations de typographie (polices de caractères, couleurs)	.docx, .odt, .html, .doc
Texte : texte <i>brut</i> sans indications de typographie. Exemple : code source ou bien des données pour un programme.	.txt, .html, .ini, .log, .conf, .c, ...

Fichier texte

Plan

1. Fichier Texte :
 1. Différence entre fichier texte brut et fichier binaire
 2. Écrire des données dans un fichier texte
 3. Lecture des données depuis un fichier texte

Fichier texte : Définition

10

Un fichier texte : [Définition]

- ▶ En informatique, un **fichier texte** ou **fichier texte brut** ou **fichier texte simple** est un fichier dont le contenu représente uniquement une suite de caractères;
- ▶ Les caractères considérés sont généralement :
 - ▶ des caractères imprimables,
 - ▶ d'espaces
 - ▶ et de retours à la ligne.



```
!"#$%&'()*+,-./  
0123456789:;<=>?  
@ABCDEFGHIJKLMNO  
PQRSTUVWXYZ[\]^_  
`abcdefghijklmnopqrstuvwxyz  
{|}~
```

Fichier texte : Définition

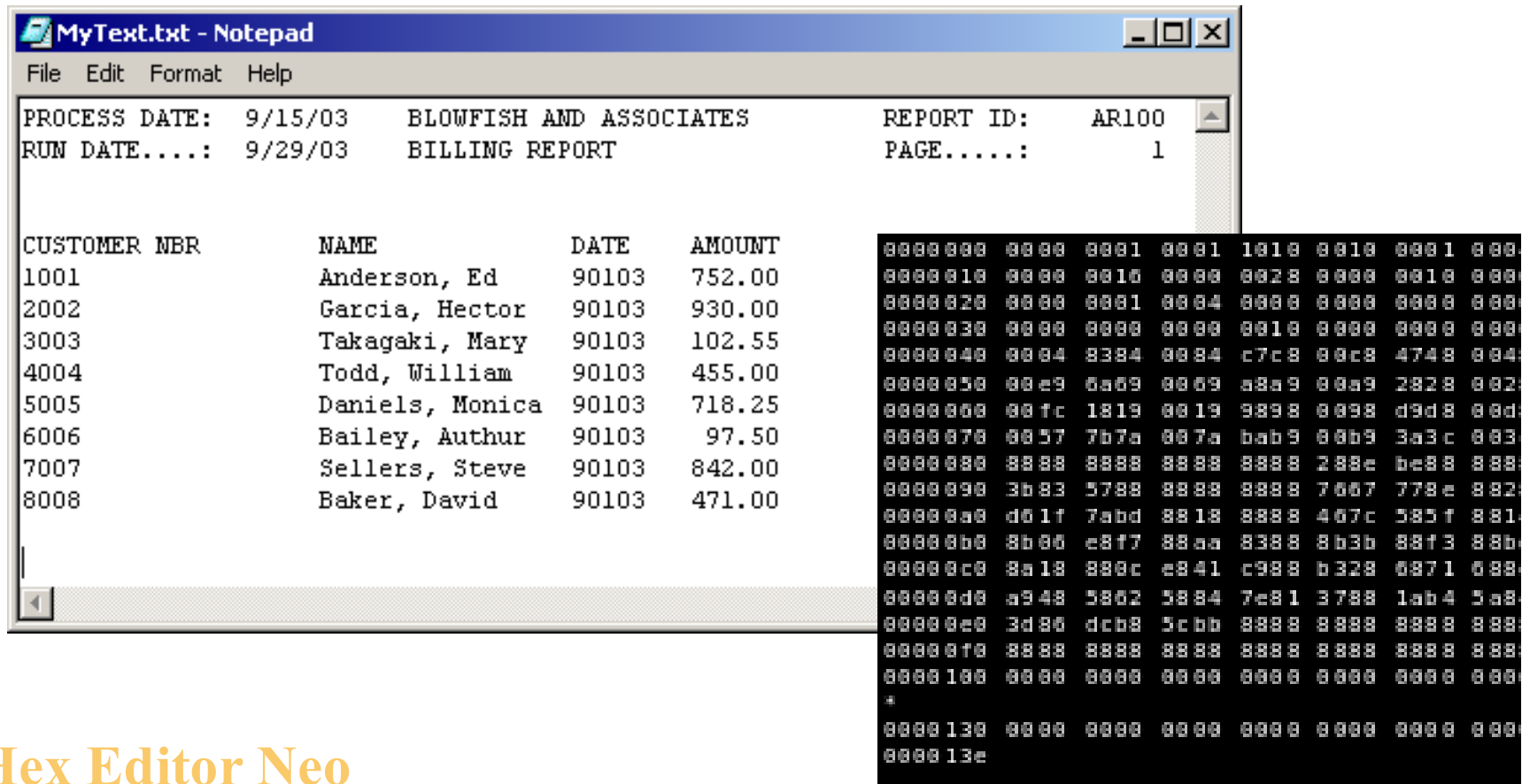
11

Un fichier binaire: [Définition]

- ▶ Un fichier « non texte » est appelé « fichier binaire », dans le sens où les bits contenus dans le fichier ne représentent pas une simple suite de caractères imprimables, d'espaces et de retours à la ligne.
- ▶ On appelle « binaire » tout fichier qui n'est pas interprétable sous forme de texte : une image, un son ou encore un fichier compressé,...
- ▶ C'est aussi un terme spécifique à l'informatique où l'on part d'un code source (texte) pour générer un logiciel sous forme de fichier exécutable (binaire).

Fichier texte : Définition

Différence entre fichier texte et fichier binaire :



Hex Editor Neo

Source : <http://www.hhdsoftware.com/free-hex-editor>

<https://sites.google.com/site/saiedanis87/>

said_anis@hotmail.com

Fichier texte : avant de commencer

13

- ▶ Au lancement de l'interpréteur, le répertoire de travail courant est celui dans lequel se trouve l'exécutable de l'interpréteur.
- ▶ Sous Windows, c'est C:\Python3X, le X étant différent en fonction de votre version de Python.
- ▶ Quand vous lancez un programme Python directement, par exemple en faisant un double-clic dessus, le répertoire courant est celui d'où vous lancez le programme.

```
>>> import os # os : operating system
>>> cur_dir = os.getcwd()# get current working directory
>>> print(cur_dir) #Afficher le répertoire courant
>>> os.chdir("C:/tests") # chdir : change directory
```

- ▶ Le répertoire **C:\tests** doit exister avant

Chemins relatifs et absolus

Les chemins absolus et relatifs sont deux moyens de décrire le chemin menant à des fichiers ou répertoires.

Pour décrire l'arborescence d'un système, on a deux possibilités :

1. les chemins absolus :

Un chemin absolu permet d'accéder à un endroit dans le disque quel que soit le répertoire de travail courant. Pour ce la, on décrit la suite des répertoires menant au fichier. Sous Windows, on partira du nom de volume (C:\, D:\...).

Exemple : C:\test\mon_fichier.txt

Remarque :

- ▶ Windows utilise des antislashes « \ » pour décrire l'arborescence de ses répertoires
- ▶ Mais, dans le code python, on utilise un slash « / » ou 2 antislashes « \\ »

Fichier texte : avant de commencer

15

2. Chemins relatifs :

- ▶ Quand on décrit la position d'un fichier grâce à un chemin relatif, cela veut dire que l'on tient compte du dossier dans lequel on se trouve actuellement.
- ▶ Quand on décrit un chemin relatif, on utilise parfois le symbole **..** (deux points) qui désigne le répertoire parent.
- ▶ Voici quelques exemples : On souhaite accéder à **fichier1.txt**
 1. Si le répertoire de travail courant est **rep2** et que, notre chemin relatif sera donc : **..\rep1\fichier1.txt**
 2. Si on se trouve dans le dossier **C:\test** notre chemin relatif sera donc : **rep1\fichier1.txt**
 3. Maintenant, si on se trouve dans **C:** notre chemin relatif sera **test\rep1\fichier1.txt**

```
■ C:
  ■ test
    ■ rep1
      ■ fichier1.txt
    ■ rep2
      ■ fichier2.txt
      ■ fichier3.txt
```

Fichier texte

Un fichier texte peut être utilisé en 3 modes

- ▶ Le mode Ecriture (mode write : w)
- ▶ Le mode Ecriture à la fin (mode append : a)
- ▶ Le mode Lecture (mode read : r)

Écrire des données dans un fichier texte

17

- ▶ Pour écrire dans un fichier, on commence par l'ouvrir *en écriture avec l'option « w » (pour « write »)*:
 - ▶ Ce qui aura pour effet de l'écraser, s'il existait déjà!
 - ▶ Ensuite, on y écrit des chaînes de caractères via la méthode **write**.
 - ▶ La méthode **write (chaîne)** n'accepte en paramètre que des chaînes de caractères et renvoie le nombre de caractères qui ont été écrits. (str() et int() pour les entiers)
 - ▶ **writelines()** permet d'écrire directement le contenu d'une liste dans le fichier.

Exemple :

```
>>> f= open ("fichier.txt", "w") # Ouvrir un fichier en écriture
>>> f.write ("Premier ligne écrite dans un fichier via Python ")
>>> f.write ("Premier ligne\n Deuxième ligne\t après 4 espaces")
>>> f.writelines (["Premier ligne\n", "Deuxième ligne\n"])
```

- ▶ Vérifier le contenu de **fichier.txt** dans le répertoire courant

Écrire des données dans un fichier texte

18

Exercice n° 1 :

Écrire un programme python qui permet de lire le nom, le prénom, la classe, le groupe, la note de test et la note de de DS de n étudiants et les enregistrer dans un fichier texte nommé notes.txt dans le répertoire c:\etudiants.

Exemple :

Nom	Prénom	Classe	Groupe	TEST	DS
BAHIJ	CHAHER	T3	A	19.00	15.50
BALLOUM	EYA	M3	A	10.50	16.00
MOHAMED	HACHEM	T4	B	14.00	16.00
ATOUI	OUMAYMA	P3	A	17.00	14.50
AMAIDI	WAFA	M4	A	17.50	18.00

Durée estimée : 10 min

On peut lire le contenu de fichier de deux manières : Lire le fichier ligne par ligne ou lire tout le fichier en bloc;

1. Lire l'intégralité de fichier : méthode **read()**

Exemple :

```
>>> f = open('fichier.txt','r') # r : mode lecture
>>> contenu = f.read() # f pointe au début de fichier
>>> print(contenu)
>>> f.close()#libérer les ressources non utilisées.
```

2. Lire le fichier ligne par ligne : méthode **readlines()**

- ▶ Soit pas à pas via la méthode **readline()**, qui lit la ligne courante et passe à la suivante,
- ▶ Soit globalement via la méthode **readlines()**, qui crée l'itérateur de toutes les lignes (liste) et permet ainsi de traiter ces dernières dans une boucle. Le contenu du fichier est donc stocké dans une liste où chaque élément de la liste (de type chaîne de caractères) correspond à une ligne du fichier.

Exemple :

```
>>> f= open('fichier.txt','r') #'r' : mode lecture
>>> print(f.readline())
>>> s = 0
>>> for ligne in f.readlines(): # méthode 1
...     s += len(ligne)
>>> for ligne in f: # méthode 2
...     print(ligne)
>>> f.close()
```

Ecrire à la fin dn fichier texte

- Pour écrire à la fin d'un fichier, on utilise la fonction open en mode ajout (append), donc contrairement au mode écriture, le contenu du fichier ne sera pas écrasé

Exemple :

```
>>> f= open('fichier.txt','a') #'a' : mode append
```

- La ligne suivante sera ajoutée au fichier sans écrasement de son contenu

```
>>> f.write("nouvelle ligne ajoutée en mode append")
```

```
>>> f.close() #fermer le fichier
```

Extraire les données d'une ligne dans des variables

22

- ▶ La méthode **split()** permet de séparer des informations dans les lignes lues selon un caractère qui lui est passé en paramètre et de retourner une liste de chaînes de caractères.

Exemple :

```
>>> 'NOM,PRENOM,Classe,DS'.split(',')#['NOM', 'PRENOM', 'Classe', 'DS']
```

- ▶ La méthode **strip()** permet d'enlever les espaces et caractères d'échappement (tabulation « \t », un retour à la ligne « \n »,...) en début et fin de ligne.

Exemple :

```
>>> ' \n \t une \t chaine\n'.strip()# en début et fin de ligne.  
'toto \t truc'  
>>> ' \n \t une \t chaine \n'.lstrip()# en début de ligne.  
'toto \t truc \n'  
>>> ' \n \t une \t chaine \n'.rstrip() # en fin de ligne.  
' \n \t toto \t truc'
```

Exercice N°2 :

Écrire un programme python qui permet d'ouvrir en lecture le fichier **notes.txt** situé dans le répertoire **c:\etudiants** et extraire les différentes informations situées dans chaque ligne du fichier dans les variables suivantes : *nom*, *prénom*, *classe*, *groupe*, *noteTest* et *noteDS* et les enregistrer dans un nouveau fichier **notes1.txt** dans lequel on écrit les mêmes données, mais formatées différemment (les données sont séparées par des *virgules* au lieu de *tabulations*).

Exemple : Contenu du fichier : notes1.txt

```
Nom,Prénom,Classe,Groupe,TEST,DS
BAHIJ,CHAHER,T3,A,19.00,15.50
BALLOUM,EYA,M3,A,10.50,16.00
MOHAMED,HACHEM,T4,B,14.00,16.00
ATOUI,OUMAYMA,P3,A,17.00,14.50
AMAIIDI,WAFA,M4,A,17.50,18.00
```

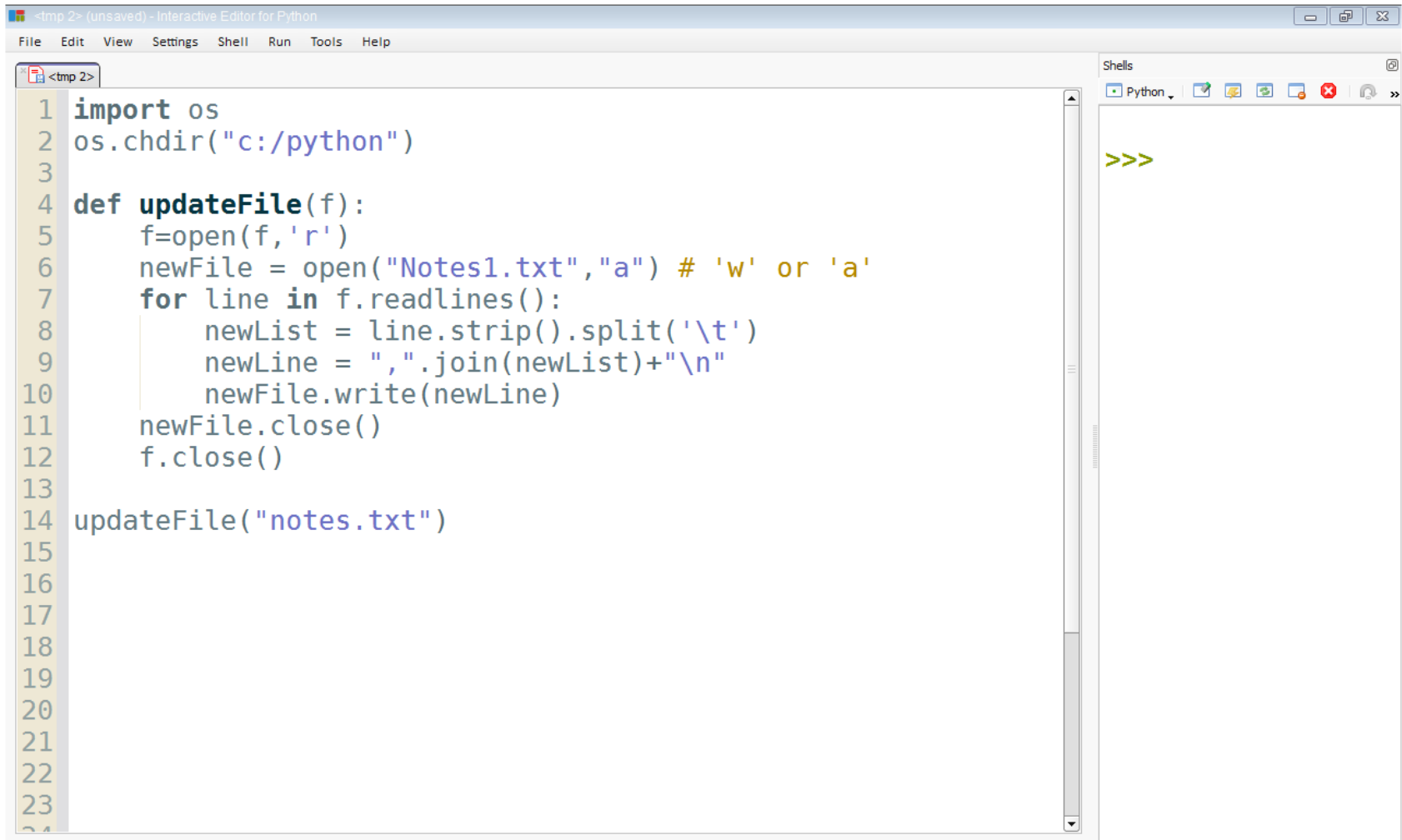
Exercise N°2 : Solution 1

```
1. import os
2. os.chdir("c:/python")

3. def updateFile(f):
4.     f=open(f, 'r')
5.     newFile = open("newNotes.txt", "a") # 'w' or 'a'
6.     for line in f.readlines():
7.         newList = line.strip().split('\t')
8.         newLine = ",".join(newList)+"\n"
9.         newFile.write(newLine)
10.    newFile.close()
11.    f.close()
12.
13. updateFile("notes.txt")
```

Exercice N°2 : Solution 1

25



The screenshot shows a Python IDE window titled "<tmp 2> (unsaved) - Interactive Editor for Python". The menu bar includes File, Edit, View, Settings, Shell, Run, Tools, and Help. The main editor area contains the following Python code:

```
1 import os
2 os.chdir("c:/python")
3
4 def updateFile(f):
5     f=open(f,'r')
6     newFile = open("Notes1.txt","a") # 'w' or 'a'
7     for line in f.readlines():
8         newList = line.strip().split('\t')
9         newLine = ",".join(newList)+"\n"
10        newFile.write(newLine)
11    newFile.close()
12    f.close()
13
14 updateFile("notes.txt")
15
16
17
18
19
20
21
22
23
24
```

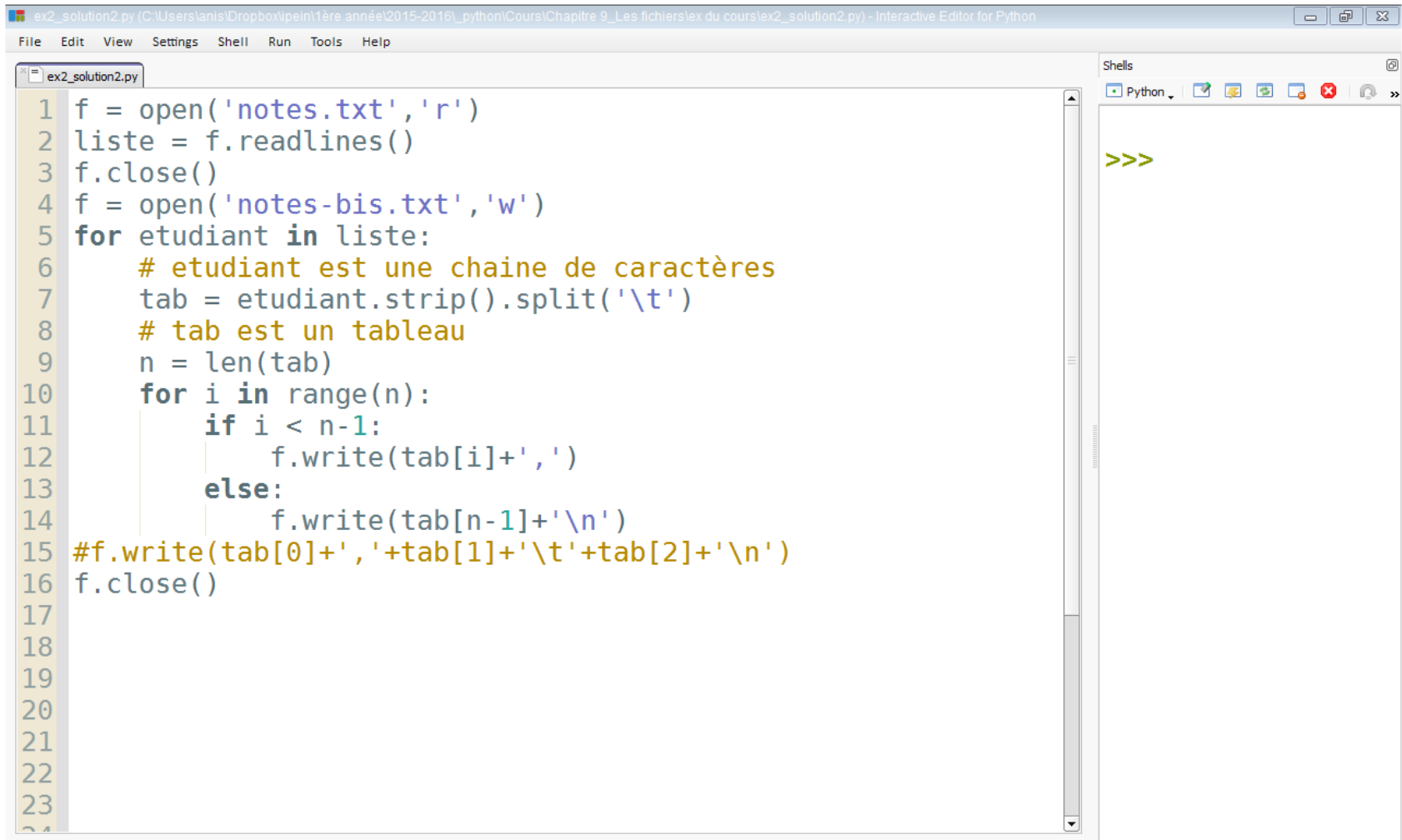
On the right side, there is a "Shells" panel with a "Python" tab. It contains a prompt symbol ">>>" indicating a ready-to-run state.

Exercice N°2 : Solution 2

```
1.  f = open('notes.txt', 'r')
2.  liste = f.readlines()
3.  f.close()
4.  f = open('notes-bis.txt', 'w')
5.  for etudiant in liste:
6.      # etudiant est une chaine de caractères
7.      tab = etudiant.strip().split('\t')
8.      # tab est un tableau
9.      n = len(tab)
10.     for i in range(n):
11.         if i < n-1:
12.             f.write(tab[i]+' , ')
13.         else:
14.             f.write(tab[n-1]+' \n')
15.     #f.write(tab[0]+' , '+tab[1]+' \t'+tab[2]+' \n')
16. f.close()
```

Exercice N°2 : Solution 2

27



```
ex2_solution2.py (C:\Users\anis\Dropbox\pein\1ère année\2015-2016\_python\Cours\Chapitre 9_Les fichiers\ex du cours\ex2_solution2.py) - Interactive Editor for Python
File Edit View Settings Shell Run Tools Help

ex2_solution2.py
1 f = open('notes.txt', 'r')
2 liste = f.readlines()
3 f.close()
4 f = open('notes-bis.txt', 'w')
5 for etudiant in liste:
6     # etudiant est une chaine de caractères
7     tab = etudiant.strip().split('\t')
8     # tab est un tableau
9     n = len(tab)
10    for i in range(n):
11        if i < n-1:
12            f.write(tab[i]+',')
13        else:
14            f.write(tab[n-1]+'\\n')
15    #f.write(tab[0]+' '+tab[1]+'\\t'+tab[2]+'\\n')
16 f.close()
17
18
19
20
21
22
23
24
```

Shells

Python >>>

- ▶ Lors de la manipulation des fichiers, il se peut qu'une erreur se produit quand on lit, quand on écrit...
- ▶ La commande **with** permet de fermer le fichier à la fin du bloc des instructions ou si une exception se produit.

Exemple :

```
>>> with open("fichier.txt","r") as f: #'r':mode lecture
...     s=f.read()
...     print(s)
```

- ▶ Cela signifie simplement que, si une exception se produit, le fichier sera tout de même fermé à la fin du bloc.
- ▶ Il est inutile, par conséquent, de fermer le fichier à la fin du bloc with. Python va le faire tout seul, qu'une exception soit levée ou non.
- ▶ Il est recommandé d'utiliser cette syntaxe, elle est plus sûre et facile à comprendre.

Fichier binaire

1. Fichier binaire :
 1. Écrire des données dans un fichier binaire
 2. Lecture des données depuis un fichier binaire

Écrire des données dans un fichier binaire

30

*Sérialisation des données avec le module **Pickle***

- Le module Pickle est extrêmement pratique pour sauvegarder dans un fichier des structures de données comme les listes (type list) ou les dictionnaires (type dict).

Exemple : sauvegarde d'un dictionnaire :

```
>>> import pickle  
  
# création d'un dictionnaire  
  
>>> coefficients = {"coeffanalyse":10,"coeffalgebe":5,"coeffchimie":5}  
  
# enregistrement du dictionnaire dans un fichier  
  
>>> f = open("data.txt","wb")#ouvrir le fichier en mode écriture binaire  
  
# sérialisation: écrire les données sous forme binaire  
  
>>> pickle.dump(coefficients,f)  
  
>>> f.close()
```

- Vérifiez le contenu du fichier data.txt !

Lire les données depuis un fichier binaire

31

*Sérialisation des données avec le module **Pickle***

- Le module Pickle est extrêmement pratique pour sauvegarder dans un fichier des structures de données comme les listes (type list) ou les dictionnaires (type dict).

Exemple : récupération de contenu du dictionnaire :

```
>>> import pickle

# récupération du dictionnaire contenu dans le fichier data.txt
>>> f = open("data.txt","rb")#ouvrir le fichier en mode lecture binaire
# désérialisation: récupérer les données sous leur forme normale
>>> coefficients = pickle.load(f)
>>> f.close()
>>> print(coefficients)
```

Fichier image

Plan

1. Fichier Image:
 1. Écrire des données dans un fichier image
 2. Lecture des données depuis un fichier image

Fichier image

1. Fichier image:

Définition

Exemple (image)

Fichier image

34

Écrire des données dans un fichier image

Fichier image

35

Lecture des données depuis un fichier image

Exercices

36

Exercice N°1:

TP : « nom du jeu »

Enoncé : fichier pdf à télécharger depuis cette adresse (saiedeanis87) ou imprimé et distribué en classe

Date limite : ..-..-..... (dernière séance)

Fichier exécutable

Ça sera la note de test