M2 ID APPRENTISSAGE
PYTHON ET MODÈLES DÉCISIONNELS COMPRÉHENSIBLES
LANGAGES DE SCRIPT (ANCIEN INTITULÉ)
TRAVAUX DIRIGÉS
Brice Mayag

TD1 : INTRODUCTION AU LANGAGE PYTHON
PARTIE 1

Ce TP est largement inspiré du cours de Python de Madame Maude Manouvrier `https://www.lamsade.dauphine.fr/~manouvri/`.

Installation et Prise en main

Cette section vous explique comment exécuter des commandes ou un programme Python depuis un éditeur Python.

Pour installer Python sur votre machine personnelle, vous devez télécharger la dernière version du langage à l'adresse `https://www.Python.org/downloads/`. L'installation de Python génère l'installation d'une interface, appelée IDLE (Python GUI). Cette interface vous permet de saisir des instructions en ligne de commande mais également d'exécuter des programmes Python enregistrés dans des fichiers. L'accès à l'interface IDLE se fait à partir du répertoire où Python a été installé. Il suffit alors de cliquer sur IDLE (voir Figure 1) qui va vous ouvrir l'interface graphique où vous pourrez taper vos instructions Python en ligne de commandes :

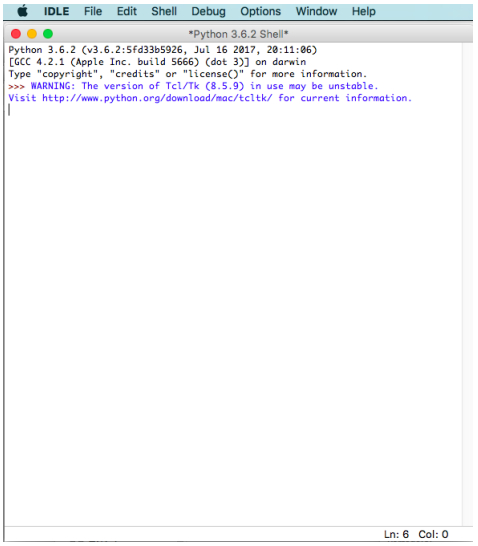

FIGURE 1 – Interface IDLE

Pour écrire un programme dans un fichier, dans le menu *File*, sélectionnez *New File*. Une nouvelle fenêtre s'ouvre. Tapez votre programme Python dans cette fenêtre (attention aux indentations). Pour exécuter votre programme, allez dans le menu Run et faites Run Modules (ou F5). Il va vous être demandé de faire une sauvegarde de votre fichier (qui a généralement l'extension *.py*), puis votre programme s'exécutera (dans la fenêtre en ligne de commande précédemment ouverte). Le début de l'execution de votre programme est indiqué dans la fenêtre en ligne de commande par le mot clé `RESTART`.

D'autres outils intégrant un éditeur de texte peuvent aussi être utilisés pour programmer en Python. Exemples : Canopy `https://www.enthought.com/products/canopy/` et IEP (Interactive Editor for Python) `http://www.pyzo.org/iep.html`.

Pour installer un package dans Canopy, il suffit d'ouvrir dans Outils le terminal Canopy (voir Figure) et de taper la commande `pip install <package name>`.

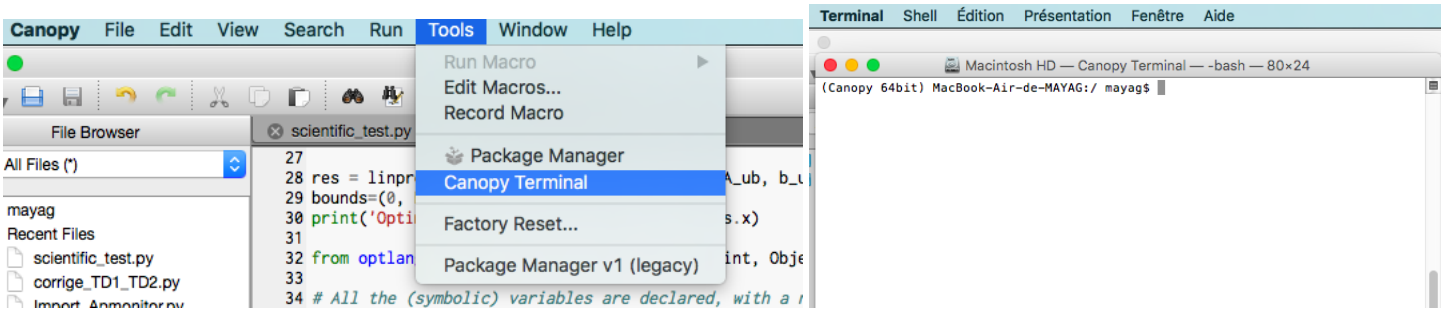

FIGURE 2 – Terminal Canopy et installation package

Premiers pas en Python

Cette section présente quelques exemples de code Python, réalisés avec `Python 3.6` en ligne de commande. Les lignes commençant par >>> correspondent aux instructions. Les lignes situées juste en dessous correspondent à l'affichage après exécution de l'instruction (i.e. après avoir tapé `<Enter>`). En Python, les commentaires commencent par le symbole #.

Vous êtes invités à taper les exemples ci-dessous pour vous entaîner et à répondre à chaque question associée aux exemples.

Exercice 1 : Quelques exemples de calcul

Essayez, en les exécutant, de comprendre ce que fait chaque instruction (non commentée) de l'exemple ci-dessous. Cet exemple est valable en ligne de commande uniquement.

```
>>> 5+3
8
>>> 5*3
15
>>> 5**3
125
>>> x=1 # declaration d'un variable x de valeur 1 (# pour le commentaire)
>>> x  # affichage de x
1
>>> a,b,c=3,5,7 # declaration de 3 variables a, b et c de valeurs resp. 3, 5 et 7
>>> a-b/c
2.2857142857142856
>>> (a-b)/c
-0.2857142857142857
>>> b/c
0.7142857142857143
>>> b//c
```

```
0
>>> b%c
5
>>> d=1.1
>>> d/c
0.15714285714285717
>>> d//c
0.0
>>> from math import * # Pour importer la librairie de fonctions mathematiques
>>> sqrt(4) # Pour calculer la racine carree
2.0
>>>pi
3.141592653589793
```

NB : La liste des fonctions de la librairie `math` est disponible à l'adresse : `http://docs.python.org/library/math.html?highlight=math#math`.

Exercice 2 : Utilisation de la fonction d'affichage print()

```
>>> print(a+b) # a et b sont les variable de l'exercice 1
8
>>> print('la valeur de', a,'+',b,'est :', a+b)
('la valeur de', 3, '+', 5, 'est :', 8)
```

Exercice 3 : Déclaration et initialisation de variables et types

```
>>> print(type(a)) # a est la variable de l'exercice 1
<class 'int'>
>>> pi=3,14
>>> print(type(pi))
<class 'tuple'>
>>> pi=3.14
>>> print(type(pi))
<class 'float'>
>>> s='exemple de chaine de caracteres'
>>> type(s)
<class 'str'>
7
>>> 2+'1.5'
Traceback (most recent call last):
File "<console>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> 2+eval('1.5') # Pour \'eliminer l'erreur pr\'ec\'edente
3.5
```

Exercice 4 : Manipulation des chaînes de caractères et exemples de fonctions sur les chaînes de caractères

```
>>> s='un exemple de chaine'
>>> s2="un autre exemple"
>>> s[1] # Acces au caractere d'indice 1 (les indices commencent a zero)
```

```
'n'
>>> print(s[0],s2[0])
u u
>>> print(s[4],s2[0])
x u
>>> print(s + ' et ' + s2) # Concatenation de chaines
un exemple de chaine et un autre exemple
>>> s3=s + ' et ' + s2
>>> s3
'un exemple de chaine et un autre exemple'
>>> s2*2
'un autre exempleun autre exemple'
>>> print('La taille de s est :', len(s))
La taille de s est : 20
>>> s3[0:3] # Recuperation des caracteres de position entre les 0 et 3e
'un '
>>> s3[4:8]
'xemp'
>>> print(s3[:3]) # Recuperation des 3 premiers caracteres
un
>>> print(s3[3:]) # Recuperation des caracteres a partir de la position 3
exemple de chaine et un autre exemple
>>> s3[::-1]
'elpmexe ertua nu te eniahc ed elpmexe nu'
>>> s3.find("exemple")
3
>>> s3.replace("chaine","str")
'un exemple de str et un autre exemple'
>>> help(str) # pour afficher l'aide
>>> sentence = 'It is raining cats and dogs'
>>> words = sentence.split()
>>> print(words)
['It', 'is', 'raining', 'cats', 'and', 'dogs']
```

Exercice 5 : Boucles et conditions

A partir de cet exercice, vous pouvez saisir vos instructions dans un fichier avec l'extension `.py` et exécuter ensuite ce fichier.

Attention : En Python il n'y a pas, comme dans certains langages, d'accolade ouvrante ou fermante pour délimiter un bloc d'instructions. Les blocs d'instructions en python sont délimités par " :" puis des tabulations : toutes les instructions consécutives à un " :" et débutant par un même nombre de tabulations appartiennent à un même bloc d'instructions.

Tapez les codes suivants et observez le résultat.

1. Boucle for

```
for i in range(10): # Ne pas oublier les deux points!!
     x = 2 # Attention ne pas oublier une tab. en debut de ligne sinon erreur!!!
     print(x*i) # Ne pas oublier la tabulation en debut de ligne!!
# Tapez encore une fois <Enter> si vous etes en ligne de commande
```

2. Boucle while

```
a=0
while(a<12): # Ne pas oublier les deux points!!
       a=a+1 # Ne pas oublier la tabulation en debut de ligne!!
       print(a, a**2,a**3) # Ne pas oublier la tabulation en debut de ligne!!
# Tapez encore une fois <Enter> si vous etes en ligne de commande
```

3. Condition If/Then/Else

```
a=0
if a==0: # Ne pas oublier les deux points!!
     print('0') # Ne pas oublier la tabulation en debut de ligne!!
elif a==1: # Ne pas mettre de tabulation et ne pas oublier les deux points!!
     print('1') # Ne pas oublier la tabulation en debut de ligne!!
else: # Ne pas mettre de tabulation et ne pas oublier les deux points!!
     print('2') # Ne pas oublier la tabulation en debut de ligne!!
# Tapez encore une fois <Enter> si vous etes en ligne de commande
```

Exercice 6 : Récupérer des saisies claviers

```
>>> s=input() # taper <Enter> puis saisir quelque chose au clavier
>>> s
>>> s=input("Saisir une chaine :") # taper <Enter>
Saisir une chaine :
>>> s
```

Récupérez, analysez et exécutez le fichier SaisieEntier.py téléchargeable sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`

Exercice 7 : Listes

Tapez chacune des insctructions suivantes (en ligne de commande) et observez le résultat.

```
>>> list=['lundi', 2, 'janvier']
>>> print(list)
>>> list[0] # Mettre print(list[0]) si vous n?etes pas en ligne de commandes
>>> list[-1]
>>> print(list[2])
>>> len(list)
>>> list.append(2010)
>>> list
>>> list[3]=list[3]+1
>>> list[3]
>>> del list[0]
>>> list
>>> list.insert(0,'mardi')
>>> list
>>> 'mardi' in list
>>> 'lundi' in list
>>> list.index("mardi")
```

```
>>> list2=list[1:3]
>>> list2
>>> list3=list[:2]
>>> list3
>>> list4=list[1:]
>>> list4
>>> list5=list[-3:-1]
>>> list5
>>> list6 = list[:-1]
>>> list6
>>> list3=list3 + [2011]
>>> list3
>>> list7=3*list
>>> list7
>>> list.extend([3,4])
>>> list
>>> list=list.pop(0)
>>> list
>>> list=[1,2,3]
>>> list2=list # Attention list et list2 correspondent a la meme
>>> list2=list.copy()#list2estunecopiedelist
>>> list.pop(1)
>>> list
>>> list2
>>> list=[1,"ab",[1,True]] # liste imbriquee
>>> list[2]
>>> print(list(range(10)))
>>> print(list(range(1,10)))
>>> print(list(range(1,10,3)))
>>> help(list) # pour l'aide sur les listes
```

Exercice 8 : Saisie d'une liste au clavier

```
>>> [input(),input(),input()]
>>> list
>>> list=[x for x in input("Saisir les elements de la liste separes
par une virgule (ex. 1,2,"abc") :").split(',')] # Taper <Enter>
>>> list
>>> list=[int(x) for x in input("Saisir des entiers separes
par une virgule (ex. 1,2,"abc") :").split(',')] # Taper <Enter>
>>> list
```

Exercice 9 : les dictionnaires

Tapez le code suivant et observez le résultat.

```
>>> dico = {} # dictionnaire vide
>>> dico['computer'] = 'ordinateur'
>>> dico['mouse'] ='souris'
>>> dico['keyboard'] ='clavier'
```

```
>>> print(dico)
>>> print(dico.keys())
>>> print(dico.values())
>>> del dico['mouse']
>>> print(dico)
>>> print(len(dico))
>>> dico.__contains__('computer')
>>> print(dico.items())
>>> for clef in dico: # Ne pas oublier les 2 points
... print(clef) # Attention ne pas oublier la tabulation!!
>>> for clef in dico: # Ne pas oublier les 2 points
... print (clef, dico[clef]) # Attention ne pas oublier la tabulation!!
>>> for clef, value in dico.items(): # Ne pas oublier les 2 points
... print(clef, value) # Attention ne pas oublier la tabulation!!
>>> dico2={'ordinateur': 'computer', 'souris'}
>>> print(dico2)
>>> dico2=dico # Attention dico et dico2 correspondent au meme dictionnaire
>>> print(dico)
>>> del dico2['computer']
>>> print(dico)
>>> print(dico2)
>>> dico3=dico.copy() # dico3 est une copie du dictionnaire reference par dico!
>>> del dico3['keyboard']
>>> print(dico)
>>> print(dico3)
>>> help(dict) # pour l'aide sur les dictionnaires
```

Exercice 10 : Références et adresses

En python, **tout est objet**, y compris les valeurs. Les variables sont des références à des valeurs.

```
>>> help(id) # pour afficher la description de la fonction retournant une adresse
>>> id(2) # on affiche l'adresse de la valeur 2
10455072
>>> a=2
>>> id(a)
10455072 # la variable 'a' a la meme adresse que celle de la valeur 2
>>> b=2
>>> id(b)
10455072 # la variable 'b' a la meme adresse que celle de la valeur 2
>>> c=b
>>> id(c)
10455072 # la variable 'c' a la meme adresse que celle de la valeur 2
>>> d=b
>>> id(d)
10455072 # la variable 'd' a la meme adresse que celle de 'b'
>>> d=3
>>> id(d)# l'adesse de 'd' a changee
10455104
>>> id(3)
```

```
10455104
>>> list=[1,2,3]
>>> id(list)
140482820775816
>>> list2=list
>>> id(list2)
140482820775816 # list et list2 ont la meme adresse
>>> id([1,2,3])
140482820774664 # Adresse de la liste [1,2,3] differente de celle de list et list2
>>> list3=[1,2,3]
>>> id(list3)
140482820772552 # adresse differente de celle de list, list2 et [1,2,3]
>>> id(list[0])
10455040
>>> id(list3[0])
10455040 # adresse de list[0] et list3[0]
>>> id(1)
10455040 # adresse de la valeur 1
>>> list is list2 # Pour tester l'identite memoire
True
>>> list[0] is 1
True
>>> list is list3
False # list et list3 correspondent a la meme liste
      # mais a 2 emplacement memoire differents
>>> list[0] is list3[0]
True
```

Exercice 11 : Définir une fonction sans paramètres

Tapez le code suivant et observez le résultat.

```
>>> def function(): # Definition de fonction sans parametre - Ne pas oublier les :
... n=10 # Faire une tabulation en debut de ligne
... while n>0: # Faire une tab. en debut de ligne et ne pas oublier les :
... print(n/2, n%2) # Faire 2 tabulations en debut de ligne
... n=n-1 # Faire 2 tabulations en debut de ligne
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> function() # Appel de la fonction
```

Exercice 12 : Définir une fonction sans paramètre qui appelle une autre fonction

Tapez le code suivant et observez le résultat.

```
>>> def function2(): # Ne pas oublier les : et une tab. sur la ligne suivante
... print('Affichage du resultat et du reste de la division des entiers de 10 \')
... function() # Faire une tabulation en debut de ligne
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> function2()
```

Exercice 13 : Définir une fonction à un paramètre

Tapez le code suivant et observez le résultat.

```
def function(n): # Definition d'une fonction a un parametre
... while n>0: # Faire une tab. en debut de ligne et ne pas oublier les :
... print(n/2, n%2) # Faire 2 tabulations en debut de ligne
... n=n-1 # Faire 2 tabulations en debut de ligne
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> function(3) # Appel de la fonction avec le parametre 3
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> function(5) # Appel de la fonction avec le parametre 5
```

Exercice 14 : Définir une fonction à deux paramètre

Tapez le code suivant et observez le résultat.

```
>>> def function(entree,diviseur): # Definition d'une fct. a plusieurs parametres
... while(entree>0):
... print(entree/diviseur, entree%diviseur)
... entree=entree-1
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> function(10,2)
>>> def afficher3fois(arg):
... print(arg, arg, arg)
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> afficher3fois(3)
>>> afficher3fois('exemple')
>>> afficher3fois([3,4])
>>> afficher3fois(3*4)
```

Exercice 15 : Valeur par défaut des paramètres

Tapez le code suivant et observez le résultat.

```
>>> def function(entree,diviseur=2): # fct. avec valeur par defaut pour diviseur
... while(entree>0):
... print(entree/diviseur, entree%diviseur)
... entree=entree-1
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> function(10)
```

Autre fonction :

```
>>> def function(entree=10,diviseur=2):
... while(entree>0):
... print(entree/diviseur, entree%diviseur)
... entree=entree-1
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> function()
>>> function(4,2)
>>> function(diviseur=4,entree=12)
```

Exercice 16 : Affecter une instance de fonction à une variable

Tapez le code suivant et observez le résultat.

```
>>> def multiplication(n,p):
...   return n*p # pour retourner une valeur
# Tapez encore une fois <Enter> si vous etes en ligne de commande
>>> a=multiplication(3,6)
>>> a
```

Exercice 17 : Fonction avec un nombre variable de paramètres

Tapez le code suivant et observez le résultat.

```
# les valeurs des arguments saisis vont etre stockes dans un t-uple - cf. print(type)
def fonction_test(*args):
    print("type de args :",type(args))
    for arg in args:
         print("paramètre:", arg)
         print()

fonction_test()
fonction_test(1)
fonction_test(1,"a")
fonction_test(1,"a",3)
```

Exercice 18 : Passage des paramètres : immuable et non immuable

En Python, il existe en effet deux types d'objets : les non immuables ou mutables tels que les listes, les dictionnaires, etc., que l'on peut modifier après leur création, et les immuables ou non mutables tels que les strings, int, floats, tuples, etc., que l'on ne peut pas modifier après leur création.

Tapez le code de cette mauvaise fonction echange dans un fichier, exécutez le et analysez le résultat.

```
def incr(a):
     print("adresse a: ",id(a))
     a=a+1
     print("Dans la fonction a =",a)

def ajouter(l):
     l=l.append(5)

a=3
print("adresse a: ",id(a))
incr(a)
print("Apres l'appel a=",a)

l=[1,2,3]
ajouter(l)
print("l=: ",l)
```

Exercice 19 : Variable locale/variable globale

Mauvaise fonction echange

```
def echange(a,b):
      print("adresses des parametres : ", id(a),id(b))
      c=a
      a=b
      b=c

x,y = 2,3
print("adresses de x et de y : ", id(x),id(y))
echange(x,y)
print("x=",x)
print("y=",y)
```

Si vous exécutez le code précédent, vous verrez que les variables `x` et `y` n'ont pas été échangées car il s'agit de variables non immuables.

Exemple de fonction echange qui fonctionne

```
def echange2(a,b):
      c=a
      a=b
      b=c
      return a,b

x,y = 2,3
x,y=echange2(x,y) # les valeurs de x et y sont bien echangees car
                  # on les modifie leur valeur ici
print("x=",x)
print("y=",y)
print() # pour sauter une ligne

a=2
b=3
a,b=echange2(a,b) # les valeurs de a et b sont bien echangees car
                  # on les modifie leur valeur ici
print("a=",a)
print("b=",b)
print()
```

Autre exemple de fonction echange qui fonctionne

```
def echangeab():
      global a # a est la variable globale du programme
      global b # b est la variable globale du programme
      c=a
      a=b
      b=c

a=2
```

```
b=3
echangeab()
print("a=",a)
print("b=",b)
```

Exercice 20 : Fonction anonyme (lambda function)

Python permet la création de fonctions anonymes (i.e. sans nom et donc non définie par `def`) à l'aide du mot clé `lambda`. Une fonction anonyme ne peut pas avoir d'instruction `return` et doit forcément retourner une expression. De telles fonctions permettent de représenter tout sous forme de fonctions sans réllement en définir explicitement.

Tapez le code suivant et observez le résultat :

```
>>> def f (x): return x**2
>>> print(f(8))
>>> g = lambda x: x**2
>>> print(g(8))
>>> (lambda x: x*2)(3) //2
>>> def make_incrementor (n): return lambda x: x + n
>>> f = make_incrementor(2)
>>> g = make_incrementor(6)
>>> print(f(42), g(42))
>>> print(make_incrementor(22)(33))
```

Exercice 21 : Fichiers

1. Instanciation du répertoire courant et modification du répertoire courant de l'interpréteur

   ```
   >>>from os import chdir
   >>>chdir("/home/login/exercices") # Mettre ici le repertoire
   # de stockage de vos exercices
   ```

   La première ligne permet d'importer la commande `chdir()` du module `os` contenant une série de fonctions permettant de dialoguer avec le système d'exploitation (os = operating system). La deuxième ligne permet de spécifier le répertoire courant de l'interpréteur python, i.e. le répertoire où il va récupérer les fichiers.

2. Manipulation de fichiers

   Tapez le code suivant et observez le résultat.

   ```
   >>> f=open('test.txt','w') # Pour ouvrir un fichier en mode ecriture
   >>> f.write("Bonjour\n") # Pour ecrire dans le fichier
   >>> f.write("Ceci est un test d'ecriture dans un fichier")
   >>> f.close() # Pour fermer le fichier
                 # verifier le fichier dans un editeur de texte
   >>> f=open('test.txt','r') # # Pour ouvrir un fichier en mode lecture
   >>> b=f.read() # Pour lire tout le fichier
   >>> print(b)
   >>> f.close() # Pour fermer le fichier
   >>> f=open('test.txt','r') # # Pour ouvrir un fichier en mode lecture
   >>> b=f.read(3) # Pour lire 3 caracteres
                   # a partir de la position courante du curseur
   >>> print(b)
   >>> f.close()
   ```

```
>>> f=open('test.txt','r')
>>> b = f.readline() # Pour lire une ligne dans le fichier
>>> print(b)
>>> b = f.readlines() # Pour lire toutes les lignes et les stocker dans une liste
>>> print(b)
>>> f.close()
>>> f=open('test.txt','a') # Pour ouvrir le fichier en mode ajout
>>> f.write("\n Fin")
>>> f.close()
>>> f=open('test.txt','r')
>>> b = f.readlines()
>>> print(len(b))
>>> print(b)
>>> f.close()
>>> f=open('test.txt','r')
>>> print(f.read()) # Pour lire tout le fichier
# deplace le curseur a la fin du fichier
>>> print(f.read())
>>> f.close()
```

3. Copie de fichiers
   Récupérez le fichier CopieFichier.py sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`. Etudiez le code en ouvrant le fichier dans un éditeur de texte et exécutez le.

4. Copier des variables dans un fichier
   Tapez le code suivant et observez le résultat.

```
>>> a = 5
>>> b = 2.83
>>> c = 67
>>> f = open('test.txt', 'w')
>>> f.write(str(a)) # Pour ecrire un entier converti en string
>>> f.write(str(b))
>>> f.write(str(c))
>>> f.close()
>>> f = open('test.txt', 'r')
>>> print(f.read())
>>> f.close()
>>> import pickle # Pour importer un module permettant
                  # de conserver le type des variables
>>> f = open('Monfichier', 'wb') # ouverture d'un fichier binaire
>>> pickle.dump(a, f) # Pour enregistrer dans le fichier
>>> pickle.dump(b, f) # equivalent a write mais en conservant le type de b
>>> pickle.dump(c, f)
>>> f.close()
>>> f = open('Monfichier', 'rb')
>>> print(f.read())
>>> f.close()
>>> f = open('Monfichier', 'rb')
>>> t = pickle.load(f) # Operation inverse de dump
>>> print(t, type(t))
```

```
>>> t = pickle.load(f) # Equivalent a read mais en conservant le type de b
>>> print(t, type(t))
>>> t = pickle.load(f)
>>> print(t, type(t))
>>> f.close()
```

Exercice 22 : Gestion des exceptions

Récupérez le fichier *ExceptionOuvertureFichier.py* téléchargeable sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`. Etudiez le code en ouvrant le fichier dans un éditeur de texte et exécutez le.

Exercice 23 : Programmation orientée-objet

1. Premier exemple de classe Récupérez le fichier *ExempleClasse1.py* téléchargeable sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`. Etudiez le code en ouvrant le fichier et exécutez le.
2. Tapez le code suivant et observez le résultat :

   ```
   >>> import ExempleClasse1 *
   >>> c1 = ExempleClasse1.CompteBancaire(?Toto?, 1000)
   >>> c1.depot(350)
   >>> c1.retrait(200)
   >>> c1.affiche()
   >>> print(c1.nom)
   >>> print(c1)
   ```

3. Accessibilité
   Reprendre la classe du point précédent et remplacer nom par `__nom`. Exécutez à nouveau l'exemple du point précédent. Observez le résultat.
4. Objet complexe Récupérez le fichier `rect_carre.py` téléchargeable sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`. Etudiez le code en ouvrant le fichier et exécutez le.
5. Héritage Récupérez le fichier `formes.py` téléchargeable sur `http://www.lamsade.dauphine.fr/~mayag/teaching.html`. Etudiez le code en ouvrant le fichier et exécutez le.
   NB : L'héritage multiple est possible en Python.