



## Langage et algorithmique CR4. Fichiers et exception

---

**A. Malek TOUMI**

**toumiab@ensta-bretagne.fr**

2017/2018

ENSTA Bretagne



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



# Principe

toggle

reset

- Mécanisme de gestion des erreurs en Python



# Principe

toggle

reset

- Mécanisme de gestion des erreurs en Python
  - Détournement du fonctionnement du programme
- ⇒ Bloc de traitement de l'erreur



# Principe

toggle

reset

- Mécanisme de gestion des erreurs en Python
  - Détournement du fonctionnement du programme
- ⇒ Bloc de traitement de l'erreur
- Nombreuses fonctions intégrées (built-in) de Python et de modules utilisent des exceptions
- ⇒ lire la documentation



## Exemple

toggle

reset

```
>>> a = 10/0
```



## Exemple

toggle

reset

```
>>> a = 10/0
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    a = 10/0
ZeroDivisionError: division by zero
>>>
```





## Exemple

```
>>> nb = int(input("Entrez un nombre: "))
```



## Exemple

```
>>> nb = int(input("Entrez un nombre: "))  
Entrez un nombre:
```



## Exemple

```
>>> nb = int(input("Entrez un nombre: "))  
Entrez un nombre: dix
```



## Exemple

```
>>> nb = int(input("Entrez un nombre: "))
Entrez un nombre: dix
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    nb = int(input("Entrez un nombre: "))
ValueError: invalid literal for int() with base 10: 'dix'
>>>
```



## Exemple

```
>>> nb = int(input("Entrez un nombre: "))
Entrez un nombre: dix
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    nb = int(input("Entrez un nombre: "))
ValueError: invalid literal for int() with base 10: 'dix'
>>>
```

- Autres exemples

- Accéder à une clé non-existante d'un dictionnaire → `KeyError`
- Chercher une valeur non-existante dans une liste → `ValueError`
- Appeler une méthode non-existante → `AttributeError`
- Référencer une variable non-existante → `NameError`
- Mélanger les types de données sans conversion → `TypeError`
- ...



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



Deux possibilités :

- traiter l'exception localement ;



Deux possibilités :

- traiter l'exception localement ;
- transmettre l'exception à la fonction appelante.





Deux possibilités :

- traiter l'exception localement ;
- transmettre l'exception à la fonction appelante.



Deux possibilités :

- traiter l'exception localement ;
- transmettre l'exception à la fonction appelante.



## Traitement local : try-except

toggle

reset

Traiter une exception localement : utiliser le bloc **try-except-finally**.

- try : certaines instructions peuvent lever une exception
- except : traitement d'une exception
- else (optionnelle). Instructions exécutées si aucune exception n'est levée
- finally (optionnelle) : instruction exécutées après le bloc **try** et les **except** éventuels.



Syntaxe :

```
try :  
    # Cette fonction peut lever une exception  
    fonction_levant_une_exception()  
# Ici on récupère toutes les exceptions  
# héritant de Exception  
except Exception as err:  
    # Traitement de l'exception  
    print (err)  
# Suite du programme
```



Traitement de plusieurs exceptions :

```
try :  
    # ...  
except Exception1 :  
    # ...  
except Exception2 :  
    # ...  
except (Exception3, Exception4) :  
    # ...  
else :  
    # ...  
finally :  
    # ...
```



## Exemple

toggle

reset

### Exemple

```
try:
    resultat = num / den
except NameError:
    print("une des variables n'a pas été définie.")
except TypeError:
    print("Type incompatible du num ou den.")
except ZeroDivisionError:
    print("denominateur est égale à 0.")
else:
    print("Le résultat obtenu est", resultat)
```



## Exemple

toggle

reset

### Exemple

```
try:
    nb = int(input("Entrez un nombre: "))
except ValueError:
    print ("Vous n'avez pas entré un nombre!")
finally:
    print("Relancer le programme")
```



## Exemple

toggle

reset

### Exemple

```
try:
    nb = int(input("Entrez un nombre: "))
except ValueError:
    print ("Vous n'avez pas entré un nombre !")
finally:
    print("Relancer le programme")
```

Entrez un nombre: dix

Vous n'avez pas entré un nombre !

Relancer le programme





## Exemple

toggle

reset

### Exemple

```
try:
    nb = int(input("Entrez un nombre: "))
except ValueError:
    print ("Vous n'avez pas entré un nombre !")
finally:
    print("Relancer le programme")
```

Entrez un nombre: dix

Vous n'avez pas entré un nombre !

Relancer le programme

Entrez un nombre: 10

Relancer le programme



## Exemple

toggle

reset

### Exemple

```
while True :  
    try:  
        nb = int(input("Entrez un nombre: "))  
        break  
    except ValueError as err:  
        print (err)  
        print ("Réessayer")
```



## Exemple

toggle

reset

### Exemple

```
while True :  
    try:  
        nb = int(input("Entrez un nombre: "))  
        break  
    except ValueError as err:  
        print (err)  
        print ("Réessayer")
```

```
Entrez un nombre: dix  
invalid literal for int() with base 10: 'dix'  
Réessayer
```



## Exemple

toggle

reset

### Exemple

```
while True :  
    try:  
        nb = int(input("Entrez un nombre: "))  
        break  
    except ValueError as err:  
        print (err)  
        print ("Réessayer")
```

```
Entrez un nombre: dix  
invalid literal for int() with base 10: 'dix'  
Réessayer  
Entrez un nombre: 10
```

## Ne pas détourner l'usage des exceptions

```
try :  
    # Calcul de l'indice désiré  
    indice = calcul_indice_tableau()  
    # Accès systématique à la liste :  
    # - l'exception gère les dépassements  
    x = tableau [indice]  
except IndexError as err:  
    pass
```

## Remarque (Remplacement de l'exception par un test)

```
# Calcul de l'indice désiré
indice = calcul_indice_tableau ()
# Test de validité de l'indice
if indice >=0 and indice <len( tableau ):
    # Accès au tableau si l'indice est valide
    x = tableau [ indice ]
```



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



# Lever une exception

toggle

reset

Lever une exception dans une méthode : utilisation de **raise**





## Lever une exception

toggle

reset

Lever une exception dans une méthode : utilisation de **raise**

Syntaxe

```
raise TypeDeLException("votre message à afficher")
```

Permet de gérer les erreurs par le mécanisme d'exception.

## Exemple (Entier positif)

```
# L'utilisateur saisit un entier
entier = input("donner un entier positif:")
try:
    # la fonction int() peut lever une exception
    entier = int(entier)
    if entier <= 0:
        # On lève une exception
        raise ValueError("nombre invalide")
except ValueError as e:
    print(e)
```

# sortie

donner un entier positif:dix

## Exemple (Entier positif)

```
# L'utilisateur saisit un entier
entier = input("donner un entier positif:")
try:
    # la fonction int() peut lever une exception
    entier = int(entier)
    if entier <= 0:
        # On lève une exception
        raise ValueError("nombre invalide")
except ValueError as e:
    print(e)
```

# sortie

donner un entier positif:dix

invalid literal for int() with base 10: 'dix'

## Exemple (Entier positif)

```
# L'utilisateur saisit un entier
entier = input("donner un entier positif:")
try:
    # la fonction int() peut lever une exception
    entier = int(entier)
    if entier <= 0:
        # On lève une exception
        raise ValueError("nombre invalide")
except ValueError as e:
    print(e)
```

# sortie

donner un entier positif:dix

invalid literal for int() with base 10: 'dix'

donner un entier positif:-10

## Exemple (Entier positif)

```
# L'utilisateur saisit un entier
entier = input("donner un entier positif:")
try:
    # la fonction int() peut lever une exception
    entier = int(entier)
    if entier <= 0:
        # On lève une exception
        raise ValueError("nombre invalide")
except ValueError as e:
    print(e)
```

# sortie

```
donner un entier positif:dix
invalid literal for int() with base 10: 'dix'
donner un entier positif:-10
nombre invalide
```



# Définition de classes d'exception

toggle

reset

- Définition de classes exception utilisateur
- Critères :
  - hériter de la classe **Exception**

## Exemple

```
class MonException (Exception ):
```



# Définition de classes d'exception

toggle

reset

- Définition de classes exception utilisateur
- Critères :
  - hériter de la classe **Exception**

## Exemple

```
class MonException (Exception ):  
    def __init__ (self, valeur ):  
        self.valeur = valeur
```



# Définition de classes d'exception

toggle

reset

- Définition de classes exception utilisateur
- Critères :
  - hériter de la classe **Exception**
  - possibilité de redéfinir `__str__`

## Exemple

```
class MonException (Exception ):  
    def __init__ (self, valeur ):  
        self.valeur = valeur  
    def __str__ ( self ):  
        return repr( self.valeur )
```





# Définition de classes d'exception

toggle

reset

- Définition de classes exception utilisateur
- Critères :
  - hériter de la classe **Exception**
  - possibilité de redéfinir `__str__`
- exception utilisable comme les autres

## Exemple

```
class MonException (Exception ):
    def __init__ (self, valeur ):
        self.valeur = valeur
    def __str__ ( self ):
        return repr( self.valeur )

try :
    raise ( MonException ("message à afficher") )
except MonException as err :
    print ('Exception : ', err.value)
```



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



# Intérêt

toggle

reset

- Sauver des données, des résultats de calcul et traitements, des simulations, ...
- Accéder à des données persistantes



- Sauver des données, des résultats de calcul et traitements, des simulations, ...
- Accéder à des données persistantes

Contrainte :

- Nécessite l'utilisation de variables pour le transfert de données mémoire/disque

Différents types de fichiers

- texte
- binaire



## Exemple de fichiers textes

- Pages Web (html, css, ...)
- Fichier journal (log), Script shell (bat)
- Images vectorielles (svg)
- Programmes Python (py)
- Les fichiers de données texte (txt, data, ...)
- Les fichiers textes formatés (xml)



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



# Lecture de fichier

toggle

reset

## Principe

- ouverture de fichier avec la fonction `open()`
- lecture de son contenu :
  - `.readline()` lit le fichier ligne par ligne
  - `.read()` lit l'ensemble du fichier
  - `.readlines()` renvoie une liste contenant chaque ligne
- fermeture de fichier avec la fonction `close()`





# Lecture de fichier

toggle

reset

## Principe

- ouverture de fichier avec la fonction `open()`
- lecture de son contenu :
  - `.readline()` lit le fichier ligne par ligne
  - `.read()` lit l'ensemble du fichier
  - `.readlines()` renvoie une liste contenant chaque ligne
- fermeture de fichier avec la fonction `close()`

## Attention

Prendre la bonne habitude de toujours fermer le fichier !



## Ouverture d'un fichier

toggle

reset

### La commande open

```
f=open("mon_fichier.txt", mode = "rt")
```



# Ouverture d'un fichier

toggle

reset

## La commande `open`

```
f=open("mon_fichier.txt", mode = "rt")
```

## Les trois modes d'accès

- **r** : ouverture en lecture (read)
- **w** : ouverture en écriture (write), le fichier est écrasé, si il n'existe pas, alors il est créé.
- **a** : ouverture en écriture en mode ajout (append), les données sont ajoutées à la suite du fichier.

avec les deux type de données :

- **"t"** pour du texte,
- **"b"** pour la lecture en mode binaire



## Exemple (Lecture des caractères d'un fichier)

```
try :
    # Ouverture du fichier texte en lecture seule
    f = open ("mon_fichier.txt") # f = open ("mon_fichier.txt","rt")
except IOError as e:
    # En cas d'exception, afficher un message d'erreur
    print ("Ouverture du fichier impossible \n", e)
else :
    # Lecture du fichier, et mise du contenu dans une variable
    caracteres = f.read ()
    print ("Nombre de caractères du fichier :", len(caracteres))
    # fermeture de fichier
    # f.close()
```



## Exemple (Lectures par lignes)

```
try :  
    f = open ("mon_fichier.txt")  
except IOError as e:  
    print ('Ouverture du fichier impossible \n', e)  
    sys.exit(1)  
# Lire le fichier sous forme de liste de lignes
```



## Exemple (Lectures par lignes)

```
try :  
    f = open ("mon_fichier.txt")  
except IOError as e:  
    print ('Ouverture du fichier impossible \n', e)  
    sys.exit(1)  
# Lire le fichier sous forme de liste de lignes  
les_lignes = f.readlines ()  
print ('Nombre de lignes :', len( les_lignes ))
```



## Exemple (Lectures par lignes)

```
try :
    f = open ("mon_fichier.txt")
except IOError as e:
    print ('Ouverture du fichier impossible \n', e)
    sys.exit(1)
# Lire le fichier sous forme de liste de lignes
les_lignes = f.readlines ()
print ('Nombre de lignes :', len( les_lignes ))
# Mettre le curseur au début du fichier
f. seek (0)
```



## Exemple (Lectures par lignes)

```
try :
    f = open ("mon_fichier.txt")
except IOError as e:
    print ('Ouverture du fichier impossible \n', e)
    sys.exit(1)
# Lire le fichier sous forme de liste de lignes
les_lignes = f.readlines ()
print ('Nombre de lignes :', len( les_lignes ))
# Mettre le curseur au début du fichier
f. seek (0)
# Lecture du fichier ligne à ligne avec une boucle for
```





## Exemple (Lectures par lignes)

```
try :
    f = open ("mon_fichier.txt")
except IOError as e:
    print ('Ouverture du fichier impossible \n', e)
    sys.exit(1)
# Lire le fichier sous forme de liste de lignes
les_lignes = f.readlines ()
print ('Nombre de lignes :', len( les_lignes ))
# Mettre le curseur au début du fichier
f. seek (0)
# Lecture du fichier ligne à ligne avec une boucle for
for ligne in f:
    # Affiche chaque ligne du fichier à l'envers
    print ( ligne [::-1] , end='')
# Fermeture du fichier après lecture
```



## Lecture de fichier

toggle

reset

### Exemple (Lectures par lignes)

```
try :
    f = open ("mon_fichier.txt")
except IOError as e:
    print ('Ouverture du fichier impossible \n', e)
    sys.exit(1)
# Lire le fichier sous forme de liste de lignes
les_lignes = f.readlines ()
print ('Nombre de lignes :', len( les_lignes ))
# Mettre le curseur au début du fichier
f. seek (0)
# Lecture du fichier ligne à ligne avec une boucle for
for ligne in f:
    # Affiche chaque ligne du fichier à l'envers
    print ( ligne[::-1] , end='')
# Fermeture du fichier après lecture
f. close ()
```



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



## Écriture d'un fichier

toggle

reset

- Méthode **write** : écrit une chaîne de caractères **str**



# Écriture d'un fichier

toggle

reset

- Méthode **write** : écrit une chaîne de caractères **str**

## Exemple

```
f=open("mon_fichier.txt","w")
f.write("Du texte")
f.write("1")
f.write("5.6")
f.write(str(46.6))
f.close()
```



# Écriture d'un fichier

toggle

reset

- Méthode **write** : écrit une chaîne de caractères **str**

## Exemple

```
f=open("mon_fichier.txt","w")
f.write("Du texte")
f.write("1")
f.write("5.6")
f.write(str(46.6))
f.close()
# contenu du fichier
Du texte15.646.6
```



## Écriture d'un fichier

toggle

reset

- Méthode **write** : écrit une chaîne de caractères **str**

### Exemple

```
f=open("mon_fichier.txt","w")
f.write("Du texte")
f.write("1")
f.write("5.6")
f.write(str(46.6))
f.close()
# contenu du fichier
Du texte15.646.6
```

### Attention

Comme il s'agit de fichier texte, la méthode **.write()** prend en argument une chaîne de caractères.



# Écriture d'un fichier

toggle

reset

## Remarques (Formatage du texte)

- Utiliser le retour à la ligne `"\n"`
- Utiliser la tabulation `"\t"`
- Formater le texte avec la méthode `.format()`

## Exemple

```
f = open("mon_fichier.txt", "wt")
f.write("ligne1 Du texte \n")
f.write("ligne2 1\n")
f.write("ligne3 \t 5.6\n")
f.write("ligne4 \t"+str(46.6)+"\n")
f.write("Ligne { :d} \t { :f} \t { :f}\n".format(5,12.566,14.5))
f.write("Ligne { :d} \t :f \t :f\n".format(6,12.666,16.5))
f.close()
```





# Écriture d'un fichier

toggle

reset

## Remarques (Formatage du texte)

- Utiliser le retour à la ligne `"\n"`
- Utiliser la tabulation `"\t"`
- Formater le texte avec la méthode `.format()`

## Exemple

```
f = open("mon_fichier.txt","wt")
f.write("ligne1 Du texte \n")
f.write("ligne2 1\n")
f.write("ligne3 \t 5.6\n")
f.write("ligne4 \t"+str(46.6)+"\n")
f.write("Ligne { :d} \t { :f} \t { :f}\n".format(5,12.566,14.5))
f.write("Ligne { :d} \t :f \t :f\n".format(6,12.666,16.5))
f.close()
```



# Écriture d'un fichier en mode append

toggle

reset

## Exemple (mode ajout 'append')

```
f = open('mon_fichier.txt', 'a')
f.write('\nUne deuxième ligne.\n') # '\n' saut de ligne
f.write('abc\tABC\t123 \n') # '\t' tabulation
f.write(str(126.85)+'\n') # str()
f.write('\x31\x41\x61\n') # écriture de '1Aa'
f.write(chr(0x62)+'\n') # écriture de 'b'
f.write(chr(99)) # écriture de 'c'
f.close()
```



# Écriture d'un fichier

toggle

reset

utilisation de la commande

## Exemple

```
# Ouverture du fichier texte en écriture
with open ( filename , mode ='w') as f: # w -> write
    for mot in [1, 'abc', (1, 2)]:
        f.write (str(mot))
        f.write ('\n')
# Fermeture du fichier implicite

# Ouverture du fichier texte en écriture
with open (filename , mode ='a') as f: # a -> append
    f.write ('Suite du fichier \n')
```



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



## Lecture/écriture

fichiers

formatés

toggle

reset

### Définition

Un fichier texte formaté représente un tableau de données où toutes les lignes présentent une structure identique.



# Lecture/écriture fichiers formatés

toggle

reset

## Définition

Un fichier texte formaté représente un tableau de données où toutes les lignes présentent une structure identique.

## Exemple

```
import numpy as np
val = np.loadtxt("fichier_formate.txt")
col = val[:,2]
np.savetxt("colonne.txt", col)
col2 = np.loadtxt("colonne.txt")
# gain en temps de chargement
```

Fichier	Edition	Format	Affichage	?
152902.00	4823.409837	00425.903959	66.804	
152903.00	4823.409837	00425.903959	66.804	
152904.00	4823.409837	00425.903959	66.804	
152905.00	4823.409837	00425.903959	66.804	



# Lecture/écriture fichiers formatés

toggle

reset

## Définition

Un fichier texte formaté représente un tableau de données où toutes les lignes présentent une structure identique.

## Exemple

```
import numpy as np
val = np.loadtxt("fichier_formate.txt")
col = val[:,2]
np.savetxt("colonne.txt", col)
col2 = np.loadtxt("colonne.txt")
# gain en temps de chargement
np.save("colonne.npy", data)
data = np.load("colonne.npy")
```

Fichier	Edition	Format	Affichage	?
152902.00	4823.409837	00425.903959	66.804	
152903.00	4823.409837	00425.903959	66.804	
152904.00	4823.409837	00425.903959	66.804	
152905.00	4823.409837	00425.903959	66.804	



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

**Fichiers binaires**

Sérialisation

## 3 Horloge





# Lecture de fichiers binaires

toggle

reset

- Un fichier est lu octet par octet
- Nécessite la connaissance de sa structure interne
- La méthode **`.read(nbOctets)`** pour lire des octets.



# Lecture de fichiers binaires

[toggle](#)[reset](#)

- Un fichier est lu octet par octet
- Nécessite la connaissance de sa structure interne
- La méthode **.read(nbOctets)** pour lire des octets.

## Remarque

- Utilisation de la méthode **.from\_bytes()** qui convertit des octets en variable entière.
- Formatage des octets lus : `struct.unpack()`  
Exemple :  
`a, b, c = struct.unpack("<BBBx",file.read(0x4)))`



# Lecture de fichiers binaires

toggle

reset

- Un fichier est lu octet par octet
- Nécessite la connaissance de sa structure interne
- La méthode `.read(nbOctets)` pour lire des octets.

## Exemple

```
# Ouverture du fichier binaire en lecture
with open(filename, mode='rb') as f:
    # Lecture du nombre de données
    octets = f.read(4)
    # Conversion du nombre de données en entier
    nb = int.from_bytes(octets, 'big')
    # nb = struct.unpack("<I",f.read(0x4)))
    for _ in range (nb):
        octets = f.read(4)
        val = int.from_bytes(octets, 'big')
        print(val)
```



# Écriture de fichiers binaires

toggle

reset

- Écriture octet par octet (ou une séquence d'octets)
- La méthode `.write()` permet d'écrire des octets dans un fichier,



# Écriture de fichiers binaires

toggle

reset

- Écriture octet par octet (ou une séquence d'octets)
- La méthode `.write()` permet d'écrire des octets dans un fichier,

## Remarque

- Utilisation de la méthode `.to_bytes()` qui convertit des entiers vers des octets.
- Formatage des octets lus : `struct.pack()`  
Exemple : `val1 = val2 = 12`  
`f.write(struct.pack("<BB", val1, val2))`



# Écriture de fichiers binaires

toggle

reset

- Écriture octet par octet (ou une séquence d'octets)
- La méthode `.write()` permet d'écrire des octets dans un fichier,

## Exemple

```
# Ouverture du fichier binaire en écriture
with open(filename, mode='wb') as f:
    size = 10 # Nombre de données
    Conversion de l'entier en liste d'octets en format "big endian"
    octets = size.to_bytes(4, 'big')
    # Écriture du nombre de données dans le fichier
    f.write(octets)
    debut = 10
    for nb in range(debut, debut + size):
        # Conversion du nombre en liste d'octets
        # Conversion sur 4 octets
        octets = nb.to_bytes(4, 'big')
        f.write(octets)
# f.write(struct.pack("<I",nb))
```



# Sommaire

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



## Principe

toggle

reset

- Sauvegarder directement des objets dans un fichier
- Puis les récupérer très simplement
- Utilisation du module **pickle**
- Lecture/écriture du fichier par **.load()/.dump()**.





## Principe

toggle

reset

- Sauvegarder directement des objets dans un fichier
- Puis les récupérer très simplement
- Utilisation du module **pickle**
- Lecture/écriture du fichier par **.load()/.dump()**.

### Attention

Ce qui a été sauvé avec pickle doit être chargé avec pickle.



# Utilisation du module pickle

toggle

reset

## Exemple

```
import pickle
# exemple d'un objet
departement = {36:'Indre',30:'Gard',75:'Paris'}
# ouverture de fichier en écriture binaire
with open('data.txt','wb') as f:
```



# Utilisation du module pickle

toggle

reset

## 1 Sauvegarder les objets

### Exemple

```
import pickle
# exemple d'un objet
departement = {36:'Indre',30:'Gard',75:'Paris'}
# ouverture de fichier en écriture binaire
with open('data.txt','wb') as f:
    pickle.dump(departement,f)
```



# Utilisation du module pickle

toggle

reset

## 1 Sauvegarder les objets

### Exemple

```
import pickle
# exemple d'un objet
departement = {36:'Indre',30:'Gard',75:'Paris'}
# ouverture de fichier en écriture binaire
with open('data.txt','wb') as f:
    pickle.dump(departement,f)
# ouverture de fichier en lecture binaire
with open('data.txt','rb') as f:
```



# Utilisation du module pickle

toggle

reset

- 1 Sauvegarder les objets
- 2 Restaurer les objets

## Exemple

```
import pickle
# exemple d'un objet
departement = {36:'Indre',30:'Gard',75:'Paris'}
# ouverture de fichier en écriture binaire
with open('data.txt','wb') as f:
    pickle.dump(departement,f)
# ouverture de fichier en lecture binaire
with open('data.txt','rb') as f:
    dep = pickle.load(f)
```



# Utilisation du module pickle

toggle

reset

## Remarques

- La méthode **.dump** (variable,fichier) permet de stocker la variable.
- Pour stocker plusieurs variables, il suffit d'appeler plusieurs fois la méthode **.dump**
- Pour lire plusieurs variables stockées, il suffit d'appeler plusieurs fois la méthode **.load**



# Sommaire

---

## 1 Les exceptions

Principe

Traitement des exceptions

Déclenchement d'une exception

## 2 Utilisation de fichiers

Principe

Lecture de fichier texte

Écriture de fichiers texte

Fichiers formatés

Fichiers binaires

Sérialisation

## 3 Horloge



## Horloge

---

toggle

reset

