

Formation au langage de programmation Python

Partie IV fonctions – manipulation de fichiers

Formateur : IBRAHIM M. S.



Global Knowledge®

du 30/05 au 02/06 2017

Chapitre : fonctions – manipulation de fichiers

1 Fonctions

- Définition
- Mécanismes d'appel
 - Arguments positionnels
 - Arguments nommés
 - Arguments optionnels
 - Nombre variable d'arguments
- Exercices

2 Manipulation de fichiers textes

- Ouverture/fermeture
- Lecture/Ecriture
- Exercices
- Fichiers binaires
- Fichiers particuliers
- Encodages

3 Résumé — questions

Utilité

- meilleure lisibilité du code, tests unitaires possibles

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommendations

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommandations

- **docstring fortement recommandée**

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommandations

- docstring fortement recommandée
- peut retourner toute type de valeur

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommandations

- docstring fortement recommandée
- peut retourner toute type de valeur
- **None** est retourné par défaut

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommandations

- docstring fortement recommandée
- peut retourner toute type de valeur
- `None` est retourné par défaut
- avoir des `return` de même type

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommandations

- docstring fortement recommandée
- peut retourner toute type de valeur
- `None` est retourné par défaut
- avoir des `return` de même type

Remarques :

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommandations

- docstring fortement recommandée
- peut retourner toute type de valeur
- `None` est retourné par défaut
- avoir des `return` de même type

Remarques :

- important : passage des arguments par référence et non par valeur

Utilité

- meilleure lisibilité du code, tests unitaires possibles
- réutilisation et maintenabilité accrue du code (non redondance)
- un pas vers la modularité : avec des bibliothèques de fonctions

Définition

```
def f_name(args_seq) :  
    """ docstring """  
  
    # f_name definition  
  
    return # statement
```

Recommandations

- docstring fortement recommandée
- peut retourner toute type de valeur
- `None` est retourné par défaut
- avoir des `return` de même type

Remarques :

- important : passage des arguments par référence et non par valeur
- `globals()` et `locals()` variables globales et locales (dictionnaires)

Arguments positionnels

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Fonctions — mécanismes d'appels

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Arguments nommés

Fonctions — mécanismes d'appels

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Arguments nommés

- arguments nommés, peuvent être donnés dans un ordre quelconque

Fonctions — mécanismes d'appels

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Arguments nommés

- arguments nommés, peuvent être donnés dans un ordre quelconque

Valeurs par défaut des arguments

Fonctions — mécanismes d'appels

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Arguments nommés

- arguments nommés, peuvent être donnés dans un ordre quelconque

Valeurs par défaut des arguments

- argument manquant remplacé par la valeur par défaut à la définition

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Arguments nommés

- arguments nommés, peuvent être donnés dans un ordre quelconque

Valeurs par défaut des arguments

- argument manquant remplacé par la valeur par défaut à la définition

Nombre d'arguments variables

Fonctions — mécanismes d'appels

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Arguments nommés

- arguments nommés, peuvent être donnés dans un ordre quelconque

Valeurs par défaut des arguments

- argument manquant remplacé par la valeur par défaut à la définition

Nombre d'arguments variables

- arguments placés dans une liste, filtrage possible des données entrées

Fonctions — mécanismes d'appels

Arguments positionnels

- arguments d'appel dans le même ordre que dans la définition

Arguments nommés

- arguments nommés, peuvent être donnés dans un ordre quelconque

Valeurs par défaut des arguments

- argument manquant remplacé par la valeur par défaut à la définition

Nombre d'arguments variables

- arguments placés dans une liste, filtrage possible des données entrées
- on peut combiner tout cela, d'autres mécanismes sont disponibles

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```


Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 2

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 2

- 1 écrire une fonction `mean` calculant la moyenne d'une liste de nombres

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 2

- 1 écrire une fonction `mean` calculant la moyenne d'une liste de nombres
- 2 écrire une fonction `mean` calculant la moyenne de ses arguments

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 2

- 1 écrire une fonction `mean` calculant la moyenne d'une liste de nombres
- 2 écrire une fonction `mean` calculant la moyenne de ses arguments

Exercice 2 – correction

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 2

- 1 écrire une fonction `mean` calculant la moyenne d'une liste de nombres
- 2 écrire une fonction `mean` calculant la moyenne de ses arguments

Exercice 2 – correction

```
1 def mean(L) : return sum(L)/len(L) if L != [] else None
```

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 2

- 1 écrire une fonction `mean` calculant la moyenne d'une liste de nombres
- 2 écrire une fonction `mean` calculant la moyenne de ses arguments

Exercice 2 – correction

- 1 `def mean(L) : return sum(L)/len(L) if L != [] else None`
- 2
 - `def mean(*a) : return (sum(a)/len(a))`

Exercice 1 : exécution et interprétation des résultats

Définition

```
def f(a,b,c=10) :  
    """ f doc """  
  
    a**b+c
```

- `f(2,31,2)`
- `f(a=2,31,2)`
- `f(2,c=31,a=2)`

- `f(2,c=31,b=2)`
- `f(c=3,a=31,b=2)`
- `f(3,2)`

Exercice 2

- 1 écrire une fonction `mean` calculant la moyenne d'une liste de nombres
- 2 écrire une fonction `mean` calculant la moyenne de ses arguments

Exercice 2 – correction

- 1

```
def mean(L) : return sum(L)/len(L) if L != [] else None
```
- 2
 - ```
def mean(*a) : return (sum(a)/len(a))
```
  - ```
def mean(*a) : L = [i for i in a] ; return sum(L)/len(L)
```


Exercice 3

- écrire une fonction mean calculant la moyenne de ses arguments

Exercice 3

- écrire une fonction mean calculant la moyenne de ses arguments
- après avoir effectué un filtrage sur les valeurs non conformes

Exercice 3

- écrire une fonction mean calculant la moyenne de ses arguments
- après avoir effectué un filtrage sur les valeurs non conformes

Exercice 4

On considère l'équation suivante :

$$a \cdot x^2 + b \cdot x + c = 0$$

- écrire une fonction retournant un tuple de la forme :

$$(n, \{x_1, \dots, x_n\})$$

- où n est le nombre de solution(s) de l'équation
- et x_i une solution de l'équation

Exercice 5 : construction d'un histogramme

```
suffrages = "buffet 707268 bové 483008 ... voynet 576666"
```

Exercice 5 : construction d'un histogramme

```
suffrages = "buffet 707268 bové 483008 ... voynet 576666"
```

- construire le dictionnaire du nombre de voix par candidats

Exercice 5 : construction d'un histogramme

```
suffrages = "buffet 707268 bové 483008 ... voynet 576666"
```

- construire le dictionnaire du nombre de voix par candidats
- écrire une fonction produisant le dictionnaire des pourcentages

Exercice 5 : construction d'un histogramme

```
suffrages = "buffet 707268 bové 483008 ... voynet 576666"
```

- construire le dictionnaire du nombre de voix par candidats
- écrire une fonction produisant le dictionnaire des pourcentages
- afficher les pourcentages associés aux candidats

Exercice 5 : construction d'un histogramme

```
suffrages = "buffet 707268 bové 483008 ... voynet 576666"
```

- construire le dictionnaire du nombre de voix par candidats
- écrire une fonction produisant le dictionnaire des pourcentages
- afficher les pourcentages associés aux candidats
- ajouter les étoiles pour obtenir un histogramme

Exercice 5 : construction d'un histogramme

suffrages = "buffet 707268 bové 483008 ... voynet 576666"

- construire le dictionnaire du nombre de voix par candidats
- écrire une fonction produisant le dictionnaire des pourcentages
- afficher les pourcentages associés aux candidats
- ajouter les étoiles pour obtenir un histogramme
- aligner visuellement les colonnes pour obtenir un affichage clair

Exercice 5 : construction d'un histogramme

suffrages = "buffet 707268 bové 483008 ... voynet 576666"

- construire le dictionnaire du nombre de voix par candidats
- écrire une fonction produisant le dictionnaire des pourcentages
- afficher les pourcentages associés aux candidats
- ajouter les étoiles pour obtenir un histogramme
- aligner visuellement les colonnes pour obtenir un affichage clair

bayrou	(18.57%) :	*****
besancenot	(4.08%) :	*****
bové	(1.32%) :	**
buffet	(1.93%) :	***
laguiller	(1.33%) :	**
le pen	(10.44%) :	*****
nihous	(1.15%) :	**
royal	(25.87%) :	*****
sarkozy	(31.18%) :	*****
schivardi	(0.34%) :	*
villiers	(2.23%) :	***
voynet	(1.57%) :	**

Fichier texte : suite d'octets représentant une suite de caractères
collection d'informations structurées stockées en mémoire de masse :

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique
- structurée : accès à l'information par indices et/ou champs

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique
- structurée : accès à l'information par indices et/ou champs

Différents modes d'ouverture d'un fichier

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique
- structurée : accès à l'information par indices et/ou champs

Différents modes d'ouverture d'un fichier

- 'r' : en lecture seule, écriture impossible, erreur si fichier inexistant

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique
- structurée : accès à l'information par indices et/ou champs

Différents modes d'ouverture d'un fichier

- 'r' : en lecture seule, écriture impossible, erreur si fichier inexistant
- 'w' : fichier ouvert en écriture, s'il existe déjà, le fichier est écrasé

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique
- structurée : accès à l'information par indices et/ou champs

Différents modes d'ouverture d'un fichier

- **'r'** : en lecture seule, écriture impossible, erreur si fichier inexistant
- **'w'** : fichier ouvert en écriture, s'il existe déjà, le fichier est écrasé
- **'a'** : en ajout, écriture à la fin du fichier, erreur si fichier inexistant

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique
- structurée : accès à l'information par indices et/ou champs

Différents modes d'ouverture d'un fichier

- **'r'** : en lecture seule, écriture impossible, erreur si fichier inexistant
- **'w'** : fichier ouvert en écriture, s'il existe déjà, le fichier est écrasé
- **'a'** : en ajout, écriture à la fin du fichier, erreur si fichier inexistant
- **'r+'** : lecture et écriture, erreur si fichier inexistant

Fichier texte : suite d'octets représentant une suite de caractères

collection d'informations structurées stockées en mémoire de masse :

- de masse : non-volatile et de grande capacité
- collection : information homogène le plus souvent
- structurée : relations bien définies entre les informations atomique
- structurée : accès à l'information par indices et/ou champs

Différents modes d'ouverture d'un fichier

- **'r'** : en lecture seule, écriture impossible, erreur si fichier inexistant
- **'w'** : fichier ouvert en écriture, s'il existe déjà, le fichier est écrasé
- **'a'** : en ajout, écriture à la fin du fichier, erreur si fichier inexistant
- **'r+'** : lecture et écriture, erreur si fichier inexistant
- **'w+'** : lecture puis écriture après suppression du contenu

Opérations : ouverture/fermeture – lecture – écriture

Opérations : ouverture/fermeture – lecture – écriture

```
● fichier = open("filename", mode='r', encoding='utf-8')
```


Opérations : ouverture/fermeture – lecture – écriture

```
• fichier = open("filename", mode='r', encoding='utf-8')  
• fichier.close()
```



Opérations : ouverture/fermeture – lecture – écriture

- `fichier = open("filename", mode='r', encoding='utf-8')`
- `fichier.close()`
- `with open("filename", mode='r') as working_file :`

Opérations : ouverture/fermeture – lecture – écriture

- `fichier = open("filename", mode='r', encoding='utf-8')`
- `fichier.close()`
- `with open("filename", mode='r') as working_file :`
  `# file manip block, file closed automatically`

Opérations : ouverture/fermeture – lecture – écriture

- `fichier = open("filename", mode='r', encoding='utf-8')`
- `fichier.close()`
- `with open("filename", mode='r') as working_file :`
  # file manip block, file closed automatically

• `str_file = fichier.read()`


totalité du fichier

Opérations : ouverture/fermeture – lecture – écriture

```
• fichier = open("filename", mode='r', encoding='utf-8')  
• fichier.close()  
• with open("filename", mode='r') as working_file :  
    ➞ # file manip block, file closed automatically
```


```
• str_file = fichier.read()          totalité du fichier  
• lst_str_lines = fichier.readlines()  totalité du fichier
```

Opérations : ouverture/fermeture – lecture – écriture

- `fichier = open("filename", mode='r', encoding='utf-8')`
- `fichier.close()`
- `with open("filename", mode='r') as working_file :`
  # file manip block, file closed automatically

- `str_file = fichier.read()` totalité du fichier
- `lst_str_lines = fichier.readlines()` totalité du fichier
- `str_file_line = fichier.readline()` une ligne du fichier

Opérations : ouverture/fermeture – lecture – écriture

- `fichier = open("filename", mode='r', encoding='utf-8')`
- `fichier.close()`
- `with open("filename", mode='r') as working_file :`
  `# file manip block, file closed automatically`

- `str_file = fichier.read()` totalité du fichier
- `lst_str_lines = fichier.readlines()` totalité du fichier
- `str_file_line = fichier.readline()` une ligne du fichier
- `for line in fichier :` chaque ligne du fichier

Opérations : ouverture/fermeture – lecture – écriture

- `fichier = open("filename", mode='r', encoding='utf-8')`
- `fichier.close()`
- `with open("filename", mode='r') as working_file :`
 ➞ # file manip block, file closed automatically

- `str_file = fichier.read()` totalité du fichier
- `lst_str_lines = fichier.readlines()` totalité du fichier
- `str_file_line = fichier.readline()` une ligne du fichier
- `for line in fichier :` chaque ligne du fichier
 ➞ # line manip block

Opérations : ouverture/fermeture – lecture – écriture

```
• fichier = open("filename", mode='r', encoding='utf-8')  
• fichier.close()  
• with open("filename", mode='r') as working_file :  
    ➞ # file manip block, file closed automatically
```

```
• str_file = fichier.read()          totalité du fichier  
• lst_str_lines = fichier.readlines()  totalité du fichier  
• str_file_line = fichier.readline()  une ligne du fichier  
• for line in fichier :               chaque ligne du fichier  
    ➞ # line manip block
```

```
• fichier.write("str_to_write")
```


Opérations : ouverture/fermeture – lecture – écriture

```
• fichier = open("filename", mode='r', encoding='utf-8')  
• fichier.close()  
• with open("filename", mode='r') as working_file :  
    ➞ # file manip block, file closed automatically
```

```
• str_file = fichier.read()          totalité du fichier  
• lst_str_lines = fichier.readlines()  totalité du fichier  
• str_file_line = fichier.readline()  une ligne du fichier  
• for line in fichier :               chaque ligne du fichier  
    ➞ # line manip block
```

```
• fichier.write("str_to_write")  
• fichier.writelines(list_of_str_lines_to_write)
```

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination
- argument optionnel : écrasement autorisé par défaut

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination
- argument optionnel : écrasement autorisé par défaut

Exercice 2 : chiffrement de César

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination
- argument optionnel : écrasement autorisé par défaut

Exercice 2 : chiffrement de César

- cryptage simple d'un fichier texte

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination
- argument optionnel : écrasement autorisé par défaut

Exercice 2 : chiffrement de César

- cryptage simple d'un fichier texte
- par décalage constant dans l'alphabet

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination
- argument optionnel : écrasement autorisé par défaut

Exercice 2 : chiffrement de César

- cryptage simple d'un fichier texte
- par décalage constant dans l'alphabet

Exercice 3 : conversion de formats

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination
- argument optionnel : écrasement autorisé par défaut

Exercice 2 : chiffrement de César

- cryptage simple d'un fichier texte
- par décalage constant dans l'alphabet

Exercice 3 : conversion de formats

- convertir un fichier csv vers le format vcf

Exercices : manipulation de fichiers

Exercice 1 : duplication de fichier

- écrire une fonction de duplication de fichier
- arguments : noms des fichiers source et destination
- argument optionnel : écrasement autorisé par défaut

Exercice 2 : chiffrement de César

- cryptage simple d'un fichier texte
- par décalage constant dans l'alphabet

Exercice 3 : conversion de formats

- convertir un fichier csv vers le format vcf
- tester le fichier produit en important les contacts

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de 'b' à la chaîne

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de 'b' à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de n octets

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice `i` par rapport à la position `n`

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice `i` par rapport à la position `n`

Pour la plupart des fichiers courants des modules dédiés existent

- `csv` : `csv`, `tablib`, `pandas`

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice `i` par rapport à la position `n`

Pour la plupart des fichiers courants des modules dédiés existent

- csv : `csv`, `tablib`, `pandas`
- excel : `openpyxl`, `xlutils`

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice `i` par rapport à la position `n`

Pour la plupart des fichiers courants des modules dédiés existent

- csv : csv, tablib, pandas
- excel : openpyxl, xlutils
- ppt : pptx, olefile, oledtools

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice `i` par rapport à la position `n`

Pour la plupart des fichiers courants des modules dédiés existent

- csv : csv, tablib, pandas
- excel : openpyxl, xlutils
- ppt : pptx, olefile, oledtools
- pdf : PyPDF2, pyfpdf, pdfcrowd

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de `'b'` à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de `n` octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice `i` par rapport à la position `n`

Pour la plupart des fichiers courants des modules dédiés existent

- csv : csv, tablib, pandas
- excel : openpyxl, xlutils
- ppt : pptx, olefile, oledtools
- pdf : PyPDF2, pyfpdf, pdfcrowd

- json : json, simplejson

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de **'b'** à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de *n* octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice *i* par rapport à la position *n*

Pour la plupart des fichiers courants des modules dédiés existent

- csv : csv, tablib, pandas
- excel : openpyxl, xlutils
- ppt : pptx, olefile, oledtools
- pdf : PyPDF2, pyfpdf, pdfcrowd

- json : json, simplejson
- doc : docx, readDocx

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de **'b'** à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de *n* octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice *i* par rapport à la position *n*

Pour la plupart des fichiers courants des modules dédiés existent

- csv : csv, tablib, pandas
- excel : openpyxl, xlutils
- ppt : pptx, olefile, oledtools
- pdf : PyPDF2, pyfpdf, pdfcrowd
- json : json, simplejson
- doc : docx, readDocx
- **svg : svgwrite, svglib, svg**

Fichier binaire : fichier que l'on lit par octet ou par bloc d'octets

- mode d'ouverture : ajout de **'b'** à la chaîne
- `content = fichier.read(n)` lecture d'un bloc de *n* octets
- `content.decode("utf-8")` bloc d'octets vers string
- `pos = fichier.tell()` renvoi la position courante du curseur
- `fichier.seek(i,n)` aller à l'indice *i* par rapport à la position *n*

Pour la plupart des fichiers courants des modules dédiés existent

- csv : csv, tablib, pandas
- excel : openpyxl, xlutils
- ppt : pptx, olefile, oledtools
- pdf : PyPDF2, pyfpdf, pdfcrowd
- json : json, simplejson
- doc : docx, readDocx
- svg : svgwrite, svglib, svg
- archives : zipfile, tarfile, gzip

La problématique des différents systèmes d'encodages

Système d'encodage utilisé par Python

```
>>> from sys import getdefaultencoding as enc_py  
>>> enc_py()           # ascii en Python 2.7.xy  
'utf-8'
```

La problématique des différents systèmes d'encodages

Système d'encodage utilisé par Python

```
>>> from sys import getdefaultencoding as enc_py
>>> enc_py()           # ascii en Python 2.7.xy
'utf-8'
```

Système d'encodage utilisé par le système

```
>>> from sys import getfilesystemencoding as enc_sy
>>> enc_sy()
'utf-8'
```

La problématique des différents systèmes d'encodages

Système d'encodage utilisé par Python

```
>>> from sys import getdefaultencoding as enc_py
>>> enc_py()           # ascii en Python 2.7.xy
'utf-8'
```

Système d'encodage utilisé par le système

```
>>> from sys import getfilesystemencoding as enc_sy
>>> enc_sy()
'utf-8'
```

now exiting Console...

- | | | | |
|--------------|-------------------|---------------|----------|
| • ISO-8859-1 | europe de l'ouest | • KO18-R | russe |
| • ISO-8859-2 | centrale, slaves | • ISO-2022-JP | japonais |
| • ISO-8859-5 | cyriliques | • HKSCS | chinois |
| • ISO-8859-6 | arabes | | |
| • ISO-8859-7 | hébreu | | |

Résumé de la séquence

Résumé de la séquence

- définition de fonction

Résumé de la séquence

- définition de fonction
- arguments positionnels

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- **NoneType et retour de fonction**

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- NoneType et retour de fonction
- type checking

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- NoneType et retour de fonction
- type checking
- **print formating**

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- NoneType et retour de fonction
- type checking
- print formating
- fichiers textes : lecture/écriture

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- NoneType et retour de fonction
- type checking
- print formatting
- fichiers textes : lecture/écriture
- **notions de systèmes d'encodages**

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- NoneType et retour de fonction
- type checking
- print formatting
- fichiers textes : lecture/écriture
- notions de systèmes d'encodages
- **manipulation de fichiers binaires**

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- NoneType et retour de fonction
- type checking
- print formatting
- fichiers textes : lecture/écriture
- notions de systèmes d'encodages
- manipulation de fichiers binaires
- exemples modules de manipulation de fichiers binaires

Résumé de la séquence

- définition de fonction
- arguments positionnels
- arguments optionnels
- arguments par défaut
- nombre variable d'arguments
- notion de docstring
- NoneType et retour de fonction
- type checking
- print formating
- fichiers textes : lecture/écriture
- notions de systèmes d'encodages
- manipulation de fichiers binaires
- exemples modules de manipulation de fichiers binaires

Questions ?