
js

J5: Objectifs

- Entrées/sorties
 - Print
 - Format
- Fichiers
- Exceptions
 - Try except
 - Raise
 - Assert
- Exercice Dico

Entrées/sorties

- input
- print
- format
- read
- readinto
- write

print

- En Python3, print devient une fonction
- sep et end sont des chaînes de caractères
- sep
 - sépare l'impression de deux arguments consécutifs
 - par défaut, un espace
- end
 - s'ajoute à la fin de la chaîne
 - par défaut, fin de ligne/retour

```
print(a, b, c, sep=",\t", end=".\n")
```

Flux d'E/S standard

- `import sys`
- `sys.stdout`
 - sortie standard
 - en général c'est l'écran
 - utilisé par `print`
- `sys.stdin`
 - entrée standard
 - en général c'est le clavier
 - utilisé par `input`
- `sys.stderr`
 - comme la sortie standard, donc l'écran
 - utilisé pour les messages d'erreur

format

- littéraux
 - chaîne introduite par f ou F
 - f-string
- %5.3f
 - format pour un nombre flottant à 5 chiffres, dont 3 après le point
- fonction format
 - notation pointée
 - substitue les {i} par les arguments de format.
- Attention : ceci s'applique à partir de 3.7

```
nom = 'Pierre'  
f'Salut {nom}'  
  
f'Pi vaut { math.pi:5.3f}'
```

```
print ('Salut {0}'.format('Pierre'))  
  
print ('Salut {nom}'.format(nom='Pierre'))
```

Fichiers

- Ouverture
 - Mode r,w,a, x: désigne l'ouverture
 - r+:mixer lecture/écriture
 - Type de fichier : t=texte , b=binaire
- Garde
 - Fermeture à la fin du gestionnaire de contexte with
- Lecture des fichiers texte
 - ligne par ligne
 - toutes les lignes dans une liste
- Lecture des fichiers binaires
 - décalage
 - pos est l'origine du décalage
 - 0:début
 - 1:relatif
 - 2:fin

```
f = open('myfile')  
data = f.read()  
f.close()
```

```
with open('myfile') as f:  
    data = f.read()  
    ...
```

```
line = f.readline()
```

```
for line in f:  
    ...
```

```
lines = f.readlines()
```

```
f.seek(offset, pos)  
l = f.read(nb_bytes)  
ba = f.readinto(nb_bytes)
```

Modes

- r Read
 - w Write
 - a Append
 - x eXclusive creation
 - + read + write
-
- t - fichier texte - lignes qui se terminent par \n (UNIX) ou \r\n (Windows) (Ascii 13 CR – Ascii 10 LF) ou \r (MacOS)
 - b – fichier binaire - pas de notion prédéterminée d'enregistrement.
 - En général, un programme qui s'arrête sans fermer un fichier ouvert en écriture fait perdre le contenu de ce fichier.
 - Autant que possible ouvrir en "r" ou en "a" protège le contenu des fichiers.



TILT

GAME OVER

THANK YOU FOR PLAYING!

Exception

- Les exceptions prédéfinies ont des noms qui se terminent par Error
- Imprévues : arrêt brutal avec l'affichage des fonctions traversées,
- Prévues : comportement d'arrêt en douceur, voire possibilité de continuer selon ce que le programmeur a conçu.
- programmation défensive

```
# Quelques exemples choisis
ArithmeticError
ImportError
NameError
OverflowError
RecursionError
ZeroDivisionError
```

Classeur Exceptions.ipynb

try

- Des erreurs se produisent à l'exécution
- On protège l'exécution d'un segment de code sensible par la clause try
- La forme complète de try
- as permet de collecter l'objet exception
 - intéressant pour faire figurer les détails dans un fichier .log

```
...  
try:  
    #section de code à protéger  
    action  
...  
except ZeroDivisionError :  
    #ce que l'on fait,  
    #si l'erreur se produit  
    action
```

```
try:  
    ...  
except xxError as ex:  
    ...  
except yyError as ey:  
    ...  
else:  
    ...# Ce que l'on fait si pas d'erreur  
finally:  
    ...#Ce que l'on fait à la fin  
    # dans tous les cas
```

raise

- lever une exception
- interrompre le cours du traitement
- pour se rattraper éventuellement sur une clause except

```
if x < 0:  
    raise Exception('Valeurs négatives  
non permises. x={}'.format(x))
```

assert

- Vérifier qu'une condition est remplie
- si ce n'est pas le cas, une exception `AssertionError` est levée
- le traitement s'arrête
- sauf si l'on est dans un cadre `try-except`.
- selon les options d'exécution (mode optimisé ou pas), les `assert` peuvent être ignorés

```
assert condition, message
```

```
try:  
    #s'assurer que a==b  
    #avant de continuer  
    assert a==b, "les objets sont  
    différents"  
    #suite du traitement  
  
except AssertionError as e:  
    logging.log(e)  
except Exception as e1:  
    #traitement de toutes les autres  
    #exceptions
```

```
$ python -O monscript.py
```


Exercice : Dico

- Traitement à mettre en place dans un classeur Jupyter
- On donne un texte écrit en Français (testinput.txt) pour effectuer la mise au point
 - Substituer les marques de ponctuation par des espaces
 - Compter les mots
 - Compter le nombre de fois où chaque mot apparaît
 - Prendre en compte l'échec possible à l'ouverture du fichier texte (un fichier de ce nom n'existe pas, ou bien on n'a pas le droit d'y accéder...) en gérant une exception
 - Donner les 10 mots les plus fréquents
 - Présenter les résultats sous la forme de tableau et graphiquement sous la forme d'un histogramme
 - Améliorer l'analyse en excluant les mots "vides" (peu de lettres, pronoms, prépositions, articles)
- Rechercher le texte intégral de "Les misérables" de Victor Hugo pour reproduire ce traitement







Trop agiles
les
stagiaires
Python !

Viens ! On se
tire !
On les aura
jamais...

Les mots-clés ajoutés

False	None	True	and	as	assert
async	await	break	class	continue	def
del	elif	else	except	finally	for
from	global	if	import	in	is
lambda	nonlocal	not	or	pass	raise
return	try	while	with	yield	

Merci !

- Restons en contact :
 - Georges Georgoulis – ggeorgoulis@alteractifs.org – 06 12 68 40 06



Coopérative d'activité et d'entrepreneurs www.alteractifs.org