



Rattrapages Python

Code Python

27 juin 2022

Version 1



Mark ANGOUSTURES <mark.angoustures@epita.fr>

Copyright

Ce document est destiné à une utilisation interne à EPITA.

Copyright © 2021/2022 Fabrice BOISSIER

La copie de ce document est soumise à conditions :

- ▷ Il est interdit de partager ce document avec d'autres personnes.
- ▷ Vérifiez que vous disposez de la dernière révision de ce document.

Table des matières

1	Consignes Générales	IV
2	Format de Rendu	V
3	Aide Mémoire	VI
4	Exercice 1 - is even	1
5	Exercice 2 - Calculatrice	3
6	Exercice 3 - Transposition de matrice	7
7	Exercice 4 - Sapin	9

1 Consignes Générales

Les informations suivantes sont très importantes :

Le non-respect d'une des consignes suivantes entraînera des sanctions pouvant aller jusqu'à la multiplication de la note finale par 0.

Ces consignes sont claires, non-ambiguës, et ont un objectif précis. En outre, elles ne sont pas négociables.

N'hésitez pas à demander si vous ne comprenez pas une des règles.

Consigne Générale 0 : Vous devez lire le sujet.

Consigne Générale 1 : Vous devez respecter les consignes.

Consigne Générale 2 : Vous devez rendre le travail dans les délais prévus.

Consigne Générale 3 : Le travail doit être rendu dans le format décrit à la section [Format de Rendu](#).

Consigne Générale 4 : Le travail rendu ne doit pas contenir de fichiers binaires, temporaires, ou d'erreurs (***~**, ***.o**, ***.a**, ***.so**, ***#***, ***core**, ***.log**, ***.exe**, binaires, ...).

Consigne Générale 5 : Dans l'ensemble de ce document, la casse (caractères majuscules et minuscules) est très importante. Vous devez strictement respecter les majuscules et minuscules imposées dans les messages et noms de fichiers du sujet.

Consigne Générale 6 : Dans l'ensemble de ce document, **login** correspond à votre login.

Consigne Générale 7 : Dans l'ensemble de ce document, **nom1-nom2** correspond à la combinaison des deux noms de votre binôme (par exemple pour Fabrice BOISSIER et Mark ANGOUSTURES, cela donnera **boissier-angoustures**).

Consigne Générale 8 : Dans l'ensemble de ce document, le caractère `_` correspond à une espace (s'il vous est demandé d'afficher `___`, vous devez afficher trois espaces consécutives).

Consigne Générale 9 : Tout retard, même d'une seconde, entraîne la note non négociable de 0.

Consigne Générale 10 : La triche (échange de code, copie de code ou de texte, ...) entraîne **au mieux** la note non négociable de 0.

Consigne Générale 11 : En cas de problème avec le projet, vous devez contacter le plus tôt possible les responsables du sujet aux adresses mail indiquées.

Conseil : N'attendez pas la dernière minute pour commencer à travailler sur le sujet.

2 Format de Rendu

Responsable(s) du projet :	Mark ANGOUSTURES <mark.angoustures@epita.fr>
Balise(s) du projet :	[RATT] [PYTHON]
Nombre d'étudiant(s) par rendu :	1
Procédure de rendu :	Devoir/Assignment sur Teams
Nom du répertoire :	login-RATT-Python
Nom de l'archive :	login-RATT-Python.tar.bz2
Date maximale de rendu :	27/06/2022 12h00
Durée du projet :	2 semaines
Architecture/OS :	Linux - Ubuntu (x86_64)
Langage(s) :	Python
Compilateur/Interpréteur :	/usr/bin/python3.8
Options du compilateur/interpréteur :	

Les fichiers suivants sont requis :

AUTHORS	contient le(s) nom(s) et prénom(s) de(s) auteur(s).
README	contient la description du projet et des exercices, ainsi que la façon d'utiliser le projet.

Votre code sera testé automatiquement, vous devez donc scrupuleusement respecter les spécifications pour pouvoir obtenir des points en validant les exercices. Votre code sera testé en appelant chaque script avec l'interpréteur python (et éventuellement un ou des arguments) :

```
python script.py arg1 arg2 [...]
```

L'arborescence attendue pour le projet est la suivante :

```
login-RATT-Python/  
login-RATT-Python/AUTHORS  
login-RATT-Python/README  
login-RATT-Python/src/  
login-RATT-Python/src/is_even.py  
login-RATT-Python/src/my_calc.py  
login-RATT-Python/src/my_pintree.py  
login-RATT-Python/src/my_transpose.py
```

3 Aide Mémoire

Le travail doit être rendu au format **.tar.bz2**, c'est-à-dire une archive **bz2** compressée avec un outil adapté (voir **man 1 tar** et **man 1 bz2**).

Tout autre format d'archive (zip, rar, 7zip, gz, gzip, ...) ne sera pas pris en compte, et votre travail ne sera pas corrigé (entraînant la note de 0).

Pour générer une archive *tar* en y mettant les dossiers *folder1* et *folder2*, vous devez taper :

```
tar cvf MyTarball.tar folder1 folder2
```

Pour générer une archive *tar* et la compresser avec GZip, vous devez taper :

```
tar cvzf MyTarball.tar.gz folder1 folder2
```

Pour générer une archive *tar* et la compresser avec BZip2, vous devez taper :

```
tar cvjf MyTarball.tar.bz2 folder1 folder2
```

Pour lister le contenu d'une archive *tar*, vous devez taper :

```
tar tf MyTarball.tar.bz2
```

Pour extraire le contenu d'une archive *tar*, vous devez taper :

```
tar xvf MyTarball.tar.bz2
```

Dans ce sujet précis, vous ferez du code en Python, qui affichera les résultats dans le terminal (donc des flux de sortie qui pourront être redirigés vers un fichier texte).

4 Exercice 1 - is even

Nom du(es) fichier(s) :	is_even.py
Répertoire :	login-RATT-Python/src/is_even.py
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est d'afficher si un nombre est pair ou impair.

Vous devez écrire un programme qui prendra un paramètre, et affichera selon ce paramètre s'il est pair ou impair.

Si le nombre est pair, vous devez écrire le mot **even** dans le terminal et renvoyer 0.

Si le nombre est impair, vous devez écrire le mot **odd** dans le terminal et renvoyer 1.

```
$ python is_even.py 3
odd
$ echo $?
1
$ python is_even.py 6
even
$ echo $?
0
$ python is_even.py 0
even
$ echo $?
0
```

Cas général

Si plusieurs ou aucun paramètre ne sont donnés, vous devez afficher le message d'erreur suivant, et renvoyer 255.

Usage: _python_is_even.py_number

```
$ python is_even.py 0 4
Usage: python is_even.py number
$ echo $?
255
$ python is_even.py
Usage: python is_even.py number
$ echo $?
255
```

Cas d'erreur 1

Si le paramètre n'est pas un nombre, vous devez écrire le mot **error** dans le terminal et renvoyer 254.

```
$ python is_even.py abcd
error
$ echo $?
254
$ python is_even.py 42a
error
$ echo $?
254
```

Cas d'erreur 2

Si plusieurs paramètres non numériques sont donnés, c'est l'erreur sur le nombre de paramètres qui primera.

```
$ python is_even.py 42 42a
Usage: python is_even.py number
$ echo $?
255
$ python is_even.py 42a 42
Usage: python is_even.py number
$ echo $?
255
```

Cas d'erreur 2

5 Exercice 2 - Calculatrice

Nom du(es) fichier(s) :	my_calc.py
Répertoire :	login-RATT-Python/src/my_calc.py
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est de créer une mini calculatrice en Python.

Vous devez écrire un programme qui prendra trois paramètres (deux nombres, puis l'opérateur), et affichera le résultat de l'opération désignée.

Vous devez implémenter les 5 opérations suivantes : l'addition (symbole **+**), la soustraction (symbole **-**), la multiplication (lettre **x**), la division (symbole **/**), et le reste de la division euclidienne (symbole **%**).

À la fin du calcul, votre programme doit renvoyer 0.

```
$ python my_calc.py 1 3 +
4
$ echo $?
0
$ python my_calc.py 144 362 -
-218
$ echo $?
0
$ python my_calc.py 6 7 x
42
$ echo $?
0
$ python my_calc.py 12 3 /
4
$ echo $?
0
$ python my_calc.py 69 2 %
1
$ echo $?
0
```

Cas général

Si des paramètres manquent, vous devez écrire le message suivant, et renvoyer 1.

Not_enough_parameters.

Usage: _python_my_calc.py_number_number_operator

Operator_might_be_: _+ _- _x _/ _%

```
$ python my_calc.py
Not enough parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
1
$ python my_calc.py 0 4
Not enough parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
1
```

Cas d'erreur 1 : pas assez de paramètres

Si des paramètres sont en trop, vous devez écrire le message suivant, et renvoyer 2.

Too_much_parameters.
Usage: _python_my_calc.py_number_number_operator
Operator_might_be_: _+_ -_x_/_%

```
$ python my_calc.py 0 4 x 45
Too much parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
2
```

Cas d'erreur 2 : trop de paramètres

Si l'opérateur n'est pas placé après les nombres/en dernier/à la troisième position, vous devez indiquer le message suivant, et renvoyer 3.

Wrong_parameters.
Usage: _python_my_calc.py_number_number_operator
Operator_might_be_: _+_ -_x_/_%

```
$ python my_calc.py x x x
Wrong parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
3
$ python my_calc.py 1 2 3
Wrong parameters.
Usage: python my_calc.py number number operator
```

```
Operator might be : + - x / %
$ echo $?
3
$ python my_calc.py 1 + 2
Wrong parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
3
```

Cas d'erreur 3 : pas les bons paramètres

Si le deuxième paramètre donné à la division est 0, vous devez renvoyer 4 et écrire le message d'erreur suivant.

Division_by_0_is_forbidden.

```
$ python my_calc.py 1 0 /
Division by 0 is forbidden.
$ echo $?
4
```

Cas d'erreur 4 : division par 0

Si plusieurs des problèmes précédents sont rencontrés simultanément, vous devez les gérer dans cet ordre de priorité : le manque de paramètres est affiché en priorité, l'excès de paramètres est affiché en seconde priorité, le mauvais ordre/mauvaise nature de paramètres en troisième, et la division par 0 en dernier.

```
$ python my_calc.py 1 0
Not enough parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
1
$ python my_calc.py 1 0 0 /
Too much parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
2
$ python my_calc.py 1 / 0
Wrong parameters.
Usage: python my_calc.py number number operator
Operator might be : + - x / %
$ echo $?
3
```

```
$ python my_calc.py 1 0 /  
Division by 0 is forbidden.  
$ echo $?  
4
```

Cas d'erreurs : ordre des erreurs

Les paramètres donnés lors des tests/de la correction seront toujours des nombres ou des opérateurs. Vous n'avez pas à gérer les cas où des caractères sont donnés en paramètres.

ATTENTION

Il est formellement interdit d'utiliser le programme **bc** ou tout autre programme ou bibliothèque de calcul.

6 Exercice 3 - Transposition de matrice

Nom du(es) fichier(s) :	my_transpose.py
Répertoire :	login-RATT-Python/src/my_transpose.py
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est de transposer une matrice.

Vous devez écrire un script nommé **my_transpose.py** qui prendra en paramètre un fichier contenant une matrice, et affichera dans le terminal la version transposée.

```
$ cat file1.txt
123
456
789
$ cat file2.txt
0123
4567
$ cat file3.txt
01
23
45
67
$ cat empty.txt
$ cat file0.txt
0
```

Exemples de fichiers d'entrée

```
$ python my_transpose.py file1.txt
147
258
369
$ python my_transpose.py file2.txt
04
15
26
37
$ python my_transpose.py file3.txt
0246
1357
$ python my_transpose.py file0.txt
0
```

Cas général

Deux cas d'erreur doivent être gérés avant tout affichage : si la matrice est vide (ou n'existe pas), et si la matrice contient autre chose que des nombres.

Si la matrice est vide ou que le fichier n'existe pas, il faut indiquer le message suivant et retourner 1.

Empty_matrix

```
$ python my_transpose.py empty.txt
Empty matrix
$ echo $?
1
$ rm -f non-existent
$ python my_transpose.py non-existent
Empty matrix
$ echo $?
1
```

Cas d'erreur 1

Si la matrice contient des caractères autres que des nombres, il faut indiquer le message suivant et renvoyer 2.

Incorrect_matrix

```
$ cat file_error.txt
123
abc
789
$ python my_transpose.py file_error.txt
Incorrect matrix
$ echo $?
2
```

Cas d'erreur 2

7 Exercice 4 - Sapin

Nom du(es) fichier(s) :	my_pintree.py
Répertoire :	login-RATT-Python/src/my_pintree.py
Droits sur le répertoire :	750
Droits sur le(s) fichier(s) :	640

Objectif : Le but de l'exercice est d'afficher un sapin en ASCII art, dont la taille varie selon le paramètre donné.

Vous devez écrire un programme qui prendra un paramètre, et affichera selon ce paramètre un sapin. Dans le cas général, affichez le sapin, et renvoyez 0.

```
$ python my_pintree.py 1
  /\
 /\ 
____
  ||
$ echo $?
0
$ python my_pintree.py 4
    /\
   /\ 
  /\ 
 /\ 
/\ 
____
  ||
$ echo $?
0
$ python my_pintree.py 5
    /\
   /\ 
  /\ 
 /\ 
/\ 
____
  ||
$ echo $?
0
```

Cas général

De façon précise, voici les spécifications pour les cas 1, 4, et 5 :

```

  /\      1 espace / 0 espace  \
 /  \    0 espace / 2 espaces \
____      4 _
||       1 espace 2 |

```

Cas général 1

```

      /\      4 espaces / 0 espace  \
     /\  \    3 espaces / 2 espaces \
    /\    \   2 espaces / 4 espaces \
   /\      \  1 espace  / 6 espaces \
  /\        \ 0 espace  / 8 espaces \
____          10 _
||           4 espaces 2 |

```

Cas général 4

```

      /\      5 espaces / 0 espace  \
     /\  \    4 espaces / 2 espaces \
    /\    \   3 espaces / 4 espaces \
   /\      \  2 espaces / 6 espaces \
  /\        \ 1 espace  / 8 espaces \
____          0 espace  / 10 espaces \
||           12 _
||           5 espaces 2 |

```

Cas général 5

Plusieurs cas spéciaux sont à prendre en compte.

Tout d'abord, dans le cas où le paramètre 0 est donné, vous retournerez 0 et afficherez :

```

$ python my_pintree.py 0
/\
||
$ echo $?
0

```

Cas 0

Si aucun ou plus de 1 paramètre est donné, vous devrez afficher le message d'erreur suivant, et retourner 1 :

Usage: `_python_my_pintree.py_number`


```
$ python my_pintree.py
Usage: python my_pintree.py number
$ echo $?
1
$ python my_pintree.py 0 4
Usage: python my_pintree.py number
$ echo $?
1
```

Cas d'erreur

Si un paramètre texte est donné, il sera interprété comme 0 :

```
$ python my_pintree.py blob
/\
||
$ echo $?
0
```

Cas texte

Si plusieurs paramètres sont donnés, qu'ils soient texte ou nombre, vous devez afficher le message d'erreur suivant et retourner 1.

```
$ python my_pintree.py plop plop
Usage: python my_pintree.py number
$ echo $?
1
$ python my_pintree.py 3 plop
Usage: python my_pintree.py number
$ echo $?
1
```

Cas texte d'erreur