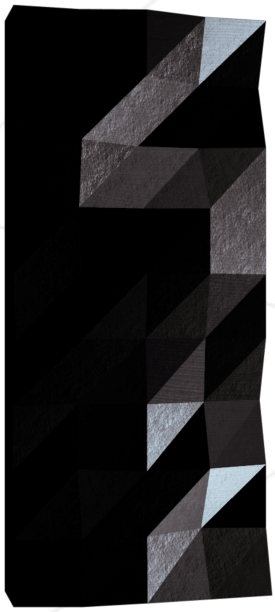

Formation Piscine Python pour la science des données - 0

Départ

Résumé : Aujourd'hui, vous apprendrez les bases du langage de programmation Python.

Version : 1.3

Contenu

Chapitre I

Règles générales

- Vous devez soumettre vos modules depuis un ordinateur du cluster ou à l'aide d'une machine virtuelle :
  - Vous pouvez choisir le système d'exploitation de votre machine virtuelle.
  - Votre machine virtuelle doit disposer de tous les logiciels nécessaires à la réalisation de votre projet. Ces logiciels doivent être installés et correctement configurés.
- Vous pouvez également utiliser directement les ordinateurs du cluster si les outils nécessaires sont disponibles.
  - Assurez-vous de disposer de suffisamment d'espace dans votre session pour installer toutes les dépendances requises pour les modules (utilisezgoinfre(si votre campus le propose).
  - Tout doit être installé avant les évaluations.
- Vos fonctions ne doivent pas s'interrompre de manière inattendue (erreur de segmentation, erreur de bus, double libération de mémoire, etc.), sauf en cas de comportement indéfini. Si un tel problème survient, votre projet sera considéré comme non fonctionnel et recevra un avertissement.0lors de l'évaluation.
- Nous vous encourageons à créer des programmes de test pour votre projet, même si ces tests **n'ont pas besoin d'être soumis et ne seront pas notés**Ces tests vous permettront d'évaluer facilement votre travail et celui de vos pairs. Ils vous seront particulièrement utiles lors de votre soutenance. Durant celle-ci, vous pourrez utiliser vos propres tests et/ou ceux du pair que vous évaluez.
- Déposez votre travail dans le dépôt Git qui vous a été attribué. Seul le travail présent dans ce dépôt sera évalué. Si Deepthought est chargé de l'évaluation, celle-ci interviendra après les évaluations par les pairs. Toute erreur détectée dans votre travail lors de l'évaluation par Deepthought entraînera l'annulation de l'évaluation.
- Vous devez utiliser la version 3.10 de Python.
- Vous pouvez utiliser n'importe quelle fonction intégrée, sauf indication contraire explicite dans l'exercice.
- Vos importations de bibliothèques doivent être explicites. Par exemple, vous devez utiliserimporter numpy comme np.Importer des bibliothèques à l'aide defrom pandas import *n'est pas autorisé et entraînera un score de0pour l'exercice.

- Les variables globales ne sont pas autorisées.
- Par Odin, par Thor ! Réfléchissez !

Chapitre II

Exercice 00

	Exercice 00
Exercice 00 : Premier script Python	
Répertoire de dépôt : *ex*00/	
Documents à remettre :Bonjour.py	
Fonctions autorisées :Aucun	

Vous devez modifier la chaîne de caractères de chaque objet de données pour afficher les messages suivants : « Bonjour le monde », « Bonjour « pays de votre campus » », « Bonjour « ville de votre campus » », « Bonjour « nom de votre campus » ».

```
ft_list=["Bonjour","tata !"] ft_tuple=(
"Bonjour","Toto !") ft_set
                ={"Bonjour","tutu!"} ={
ft_dict         "Bonjour":"titi !"}

# Votre code ici

imprimer(ft_list)
imprimer(ft_tuple)
imprimer(ft_set)
imprimer(ft_dict)
```

Résultat attendu :

```
$>python Hello.py | cat -e
['Hello', 'World!']$
(« Bonjour, France ! »)
{'Bonjour', 'Paris!'}$
{'Bonjour': '42Paris!'}$
$>
```

Chapitre III

Exercice 01

	Exercice 01
Exercice 01 : Première utilisation du paquet	
Répertoire de dépôt : *ex*01/	
Documents à remettre :format_ft_time.py	
Fonctions autorisées :heure, datetime ou toute autre bibliothèque permettant de recevoir la date	

Écrivez un script qui formate les dates de cette manière. Bien sûr, votre date ne sera pas exactement la même que la mienne, comme dans l'exemple, mais le formatage doit être identique.

Résultat attendu :

```
$>python format_ft_time.py | cat -e
Secondes écoulées depuis le 1er janvier 1970 : 1 666 355 857,3622 ou 1,67e+09 en notation scientifique$ 21
oct. 2022$
$>
```

Chapitre IV

Exercice 02

	Exercice 02
Exercice 02 : Première fonction Python	
Répertoire de dépôt : *ex*02/	
Documents à remettre :trouver_ft_type.py	
Fonctions autorisées :Aucun	

Écrivez une fonction qui affiche les types d'objets et renvoie 42.

Voici comment il devrait être prototypé :

```
def tout_chose_est_objet(objet:n'importe lequel)->int:
        # Votre code ici
```

Votre fichier tester.py :

```
from find_ft_type import all_thing_is_obj

ft_list=["Bonjour","tata !"] ft_tuple=(
"Bonjour","Toto !") ft_set
        ={"Bonjour","tutu!"} ={
ft_dict  "Bonjour":"titi !"}

tout_objet_est_un_objet(ft_list)
tout_objet_est_un_objet(ft_tuple)
tout_objet_est_un_objet(ft_set)
tout_objet_est_un_objet(ft_dict)
tout_objet_est_un_objet('Brian")
tout_est_objet("Toto")
print(tout_est_objet(10))
```

Résultat attendu :

```
$>python tester.py | cat -e List :
<class 'list'>$ Tuple : <class
'tuple'>$ Set : <class 'set'>$

Dictionnaire : <class 'dict'>$
Brian est dans la cuisine : <class 'str'>$ Toto est
dans la cuisine : <class 'str'>$ Type introuvable$

42$
$>
```

L'exécution de votre fonction seule ne produit aucun effet.

Résultat attendu :

```
$>python find_ft_type.py | cat -e $>
```

Chapitre V

Exercice 03

	Exercice 03
Exercice 03 : NULL introuvable	
Répertoire de dépôt : *ex*03/	
Documents à remettre :NULL_not_found.py	
Fonctions autorisées :Aucun	

Écrivez une fonction qui affiche le type de tous les objets de type « Null ».
Retournez 0 en cas de succès et 1 en cas d'erreur.
Votre fonction doit afficher tous les types de « Null ».

Voici comment il devrait être prototypé :

```
def NULL_not_found(objet:n'importe lequel)->int:
  # Votre code ici
```

Votre fichier tester.py :

```
from NULL_not_found import NULL_not_found

Rien=Aucun
Ail       =flotter("NaN") =
Zéro      0
Vide      =""
Faux      =FAUX

NULL_not_found(Rien)
NULL_not_found(Ail)
NULL_not_found(Zéro)
NULL_not_found(Vide)
NULL_not_found(Faux)
imprimer(NULL_not_found("Brian"))
```

Résultat attendu :

```
$>python tester.py | cat -e Nothing:
None <class 'NoneType'>$ Cheese: nan
<class 'float'>$ Zero: 0 <class 'int'>$

Vide : <class 'str'>$
Faux : False <class 'bool'>$ Type
introuvable$
1$
$>
```

L'exécution de votre fonction seule ne produit aucun effet.

Résultat attendu :

```
$>python NULL_not_found.py | cat -e $>
```

Chapitre VI

Exercice 04

	Exercice 04
Exercice 04 : Les nombres pairs et impairs	
Répertoire de dépôt : *ex*04*/*	
Documents à remettre :whatis.py	
Fonctions autorisées :sys ou toute autre bibliothèque permettant de recevoir les arguments	

Créez un script qui prend un nombre comme argument, vérifie s'il est pair ou impair, et affiche le résultat.

Si plusieurs arguments sont fournis ou si l'argument n'est pas un entier, affichez un **Erreur d'assertion**.

Résultat attendu :

```
$> python whatis.py 14 Je
suis égal.
$>
$> python whatis.py -5 Je
suis bizarre.
$>
$> python whatis.py
$>
$> python whatis.py 0 Je
suis égal.
$>
$> python whatis.py Salut ! AssertionError :
l'argument n'est pas un entier $>

$> python whatis.py 13 5
AssertionError : plusieurs arguments ont été fournis $>
```

Chapitre VII

Désormais, vous devez respecter ces règles supplémentaires.

- Aucun code dans la portée globale. Utilisez des fonctions !
- Chaque programme doit avoir une fonction principale et ne pas être un simple script :

```
def main():
    # vos tests et votre gestion des erreurs

si __name__ == "__main__":
    principal()
```

- Toute exception non interceptée invalidera les exercices, même en cas d'erreur que vous étiez censé tester.
- Toutes vos fonctions doivent avoir une documentation (__doc__).
- Votre code doit respecter la norme
    - pip install flake8
    - alias norminette=flake8

Chapitre VIII

Exercice 05

	Exercice 05
Exercice 05 : Premier programme autonome en Python	
Répertoire de dépôt : *ex*05/	
Documents à remettre :construction.py	
Fonctions autorisées :sys ou toute autre bibliothèque permettant de recevoir les arguments	

Cette fois, vous devez créer un véritable programme autonome, avec une fonction principale, qui prend une seule chaîne de caractères en argument et affiche la somme de ses caractères majuscules, minuscules, de sa ponctuation, de ses chiffres et de ses espaces.

- Si aucune information n'est fournie, l'utilisateur est invité à saisir une chaîne de caractères.
- Si plusieurs arguments sont fournis au programme, affichez un**Erreur d'assertion**.

Résultats attendus :

```
$>python building.py "Python 3.0, sorti en 2008, était une révision majeure qui n'est pas complètement rétrograde
        Compatible avec les versions antérieures. Python 2 a été abandonné avec la version 2.7.18 en 2020. (Ce
texte contient 171 caractères.)
2 lettres majuscules
121 lettres minuscules
7 signes de ponctuation
26 places
15 chiffres
$>
```

Résultats attendus : (le retour chariot compte comme un espace ; si vous ne souhaitez pas en insérer un, utilisez Ctrl + D)

```
$>python building.py
Quel est le texte à compter ?
Bonjour le monde !
Le texte contient 13 caractères : 2
lettres majuscules
8 lettres minuscules
1 signe de ponctuation
2 espaces
0 chiffres
$>
```

Par Odin, par Thor ! Réfléchissez ! Ne réinventez pas la roue, utilisez les fonctionnalités du langage.

Chapitre IX

Exercice 06

	Exercice 06
Exercice 06 :	
Répertoire de dépôt : *ex*06/	
Documents à remettre :ft_filter.py, filterstring.py	
Fonctions autorisées :sys ou toute autre bibliothèque permettant de recevoir les arguments	

Partie 1 : Fonction de filtrage de recodage

Recodez votre propre fonction ft_filter ; elle devrait se comporter comme la fonction intégrée d'origine.
(cela devrait renvoyer la même chose que "print(filter.__doc__)"), vous devriez utiliser**compréhension de listes**pour recoder votre ft_filter.

Bien entendu, l'utilisation du filtre intégré d'origine est interdite.

Vous pouvez valider le module à partir d'ici, mais nous vous encourageons à poursuivre car vous aurez besoin de certaines informations pour les projets suivants.

Partie 2 : Le programme

Créez un programme qui accepte deux arguments : une chaîne de caractères (S) et un entier (N). Le programme doit afficher une liste de mots issus de**S**qui ont une longueur supérieure à**N**.

• Les mots sont séparés les uns des autres par des espaces.

• Les chaînes de caractères ne contiennent aucun caractère spécial (ponctuation ou invisible).

• Le programme doit contenir au moins un**compréhension de liste**expression et une **lambda**.

• Si le nombre d'arguments est différent de 2, ou si le type d'un argument est incorrect, le programme affiche un message d'erreur.**Erreur d'assertion**.

Résultats attendus :

```
$> python filterstring.py 'Hello the World' 4 ['Hello',
'World']
$>
```

```
$> python filterstring.py 'Hello the World' 99 []

$>
```

```
$> python filterstring.py 3 'Hello the World'
AssertionError: les arguments sont incorrects $>
```

```
$> python filterstring.py AssertionError : les
arguments sont incorrects $>
```

Chapitre X

Exercice 07

	Exercice 07
Exercice 07 : Dictionnaires SoS	
Répertoire de dépôt : *ex*07*/*	
Documents à remettre :sos.py	
Fonctions autorisées :sys ou toute autre bibliothèque permettant de recevoir les arguments	

Créez un programme qui prend une chaîne de caractères comme argument et l'encode enCode Morse.

- Le programme prend en charge les espaces et les caractères alphanumériques.
- Un caractère alphanumérique est représenté par des points (.) et des tirets (-).
- Les caractères Morse complets sont séparés par un seul espace.
- Un espace est représenté par une barre oblique /.

Vous devez utiliser un**dictionnaire**pour stocker votre code Morse.

```
NESTED_MORSE={      ".":"/ ",
                    "UN":".- ",
                    . . .
```

Si le nombre d'arguments est différent de 1, ou si le type d'un argument est incorrect, le programme affiche une erreur.**Erreur d'assertion**.

```
$> python sos.py "sos" | cat -e . . . - -
- . . . $
$> python sos.py 'h$llo' AssertionError: les
arguments sont incorrects $>
```

Chapitre XI

Exercice 08

	Exercice 08
Exercice 08 : Chargement...	
Répertoire de dépôt : *ex*08/	
Documents à remettre :Chargement.py	
Fonctions autorisées :os	

Créons donc une fonction appeléeft_tqdm.
La fonction doit copier la fonction tqdm avec lerendementopérateur.

Voici comment il devrait être prototypé :

```
def ft_tqdm(lst:gamme)->Aucun:
  # Votre code ici
```

Votre fichier tester.py : (vous comparez votre version avec l'originale)

```
from time import sleep from
tqdm import tqdm from Loading
import ft_tqdm

pourélément dans ft_tqdm(plage(333)):
        dormir(0,005)
imprimer()
pourélément dans tqdm(plage(333)):
        dormir(0,005)
imprimer()
```

Résultat attendu : (vous devez obtenir une fonction aussi proche que possible de la version originale)

```
    $> python tester.py
100%|[============================================================>]| 333/333 100%|
                                                                    | 333/333 [00:01<00:00, 191,61it/s]
```

Vous pouvez utiliser get_terminal_size pour adapter la taille à celle de votre terminal.

Chapitre XII

Exercice 09

	Exercice 09
Exercice 09 : Ma première création de paquet	
Répertoire de dépôt : *ex*09/	
Fichiers à remettre : *.py, *.txt, *.toml, README.md, LICENCE	
Fonctions autorisées :PyPI ou toute autre bibliothèque pour la création de paquets	

Créez votre premier paquet Python comme vous le souhaitez ; il apparaîtra dans la liste des paquets installés lorsque vous taperez la commande « pip list » et ses caractéristiques s'afficheront lorsque vous taperez « pip show -v ft_package ».

```
$>pip show -v ft_package
Nom : ft_package
Version : 0.0.1
Résumé : Un exemple de package de test
Page d'accueil : https://github.com/eagle/ft_package
Auteur : eagle
Auteur-email : eagle@42.fr
Licence : MIT
Emplacement : /home/eagle/...
Prérequis :
Requis par :
Version des métadonnées : 2.1
Programme d'installation : pip
Classificateurs :
Points d'entrée :
$>
```

Le paquet sera installé via pip en utilisant l'une des commandes suivantes (les deux devraient fonctionner) :

- pip install ./dist/ft_package-0.0.1.tar.gz
- pip install ./dist/ft_package-0.0.1-py3-none-any.whl

Votre package doit pouvoir être appelé depuis un script comme celui-ci :

```
from ft_package import count_in_list

imprimer(compte_dans_liste(["toto","tata","toto"],"toto")) # sortir: 2
imprimer(compte_dans_liste(["toto","tata","toto"],"tutu")) # sortir: 0
```

Chapitre XIII

Soumission et évaluation par les pairs

Remettez votre devoir dans votreGitLe dépôt reste inchangé. Seul le contenu de votre dépôt sera évalué lors de la soutenance. N'hésitez pas à revérifier l'exactitude des noms de vos dossiers et fichiers.

Le processus d'évaluation se déroulera sur l'ordinateur du groupe évalué.