Formation Python – Livret d'exercices

Table des matières

Exo 2 : les nombres

1 – Additionnez 2 valeurs entières et affecté le résultat à une variable nommée `res`

2 – Multipliez la variable `res` par 3.14159 et affecté le résultat à une variable nommée `fres`. De quel type est cette variable ?

3 – Ajoutez 1 à la valeur booléenne True. Afficher le résultat. Que se passe t il ? Qu'en déduisez vous ?

4 – Diviser 10.4 par None. Que se passe t il ? Pourquoi ?

5 – Quel est le type de None ?

6 – Est ce que l'expression 2j + 4 est valide en Python ? A quoi cela correspond il ?

Exo 4 : les chaînes de caractères

Voici une chaîne de caractère : s = 'hello la compagnie'

1 – Quelle est la longueur de la chaîne s ?

2 – Quelle est la position du caractère 'i' dans s ?

3 – Afficher les 3 derniers caractères de s ?

4 – Afficher la chaîne s en majuscule ?

5 – Est ce que la chaîne s commence par 'hell' ?

6 – Compter le nombre de caractère 'e' dans s.

7 – Est ce que la chaîne 'o l' est dans s ?

8 – Est ce que la chaîne s est de type 'str' ?

9 – Pourquoi l'instruction suivante provoque une erreur ?

```
s[0] =  'e'
```

10 - Afficher la sous-chaîne de s qui part du 3° caractère au 8° caractère non compris en ne prenant que un caractère sur deux.

Exo 6 : les listes

Soit la liste l = [12, 6.14, True, 'hello'], et la liste ll = [False, (12, 23)]

Trouver l'unique opération correspondante à chaque nouvelle valeur de l

1 – l = [12, 6.14, True, None, 'hello']
2 – l = [120, 6.14, True, None, 'hello']
3 – l = [6.14, 120, True, None, 'hello'] [1]
4 – l = [6.14, 120, None, 'hello']
5 – l = ['hello', None, 120, 6.14]
6 – l = ['HELLO', None, 120, 6.14]
7 – l = ['HELLO', None, 120, 6.14, False, (12, 23)]

1 En Python l'échange de 2 valeurs peut s'écrire `a, b = b, a`

Exo 8 : les dictionnaires

Soit les dictionnaires d1 = {'x' : 3, 'y' : 5, 'z' : -10.7} , d2 = { 'u':10, 'v' : 4 }

Trouver l'unique opération correspondante à chaque nouvelle valeur de d1 et/ou d2

1 – d1 = {'x': 3, 'y': 5, 'z': 0}
2 – d1 = {'x': 3, 'z': 0}
3 – d1 = {'x': 3, 'y':15, 'z': 0}
4 – d1 = {'x': 9, 'y':15, 'z': 0}
5 – d1 = {'x': 9, 'y':15, 'z': 0, 'l':(1, 3 , 7)}
6 – d1 = {'x': 9, 'y':15, 'z': 0, 'l':(1, 3 , 7), 'u':10, 'v': 4}

Exo 9 : les ensembles

Soit l'ensemble suivant : s1 = {10, 20, 40, 50, 60}

1 - Ajouter la valeur entière 10

2 – Supprimer la valeur 60

3 – Trouver les éléments communs entre s1 et s2 = {30, 40, 10}

4 – Trouvez les éléments qui sont exclusivement dans l'un ou l'autre ensemble

5 – Indiquez si s3 = {10, 50} est un sous-ensemble de s1.

Exo 10 : les structures conditionnelles

1 - Ecrire en Python une série d'instructions qui permet afficher à l'écran le maximum entre 2 variables x et y

2 – Ecrire en Python une série d'instructions qui permet afficher à l'écran le maximum entre 3 variables x, y et z

Exo 11 : la boucle `while`

Ecrire une fonction en Python qui affiche à l'écran en majuscule une valeur saisie depuis le clavier. La fonction interne qui permet de lire depuis le clavier est 'input('Tapez votre texte puis presser la touche enter')'. Si la chaîne 'QQ' est lue depuis le clavier le programme s'arrêtera.

Exo 12 : la boucle `for`

Ecrire une fonction fizz_buzz en Python qui pour les `n` premiers entiers positifs affiche à l'écran :

- 'fizz' si la valeur est un multiple de 3
- 'buzz' si la valeur est un multiple de 5
- 'fizzbuzz' si la valeur est un multiple de 3 et de 5
- la valeur quand aucune des restrictions précédentes ne s'applique

Pour fizz_buzz(15), nous aurons :

1, 2, fizz, 4, buzz, fizz, 7, 8, fizz, buzz , 11, fizz, 13, 14, fizzbuzz

N sera passé en paramètre à la fonction. La fonction interne python 'range' permet de récupérer via une boucle les nombres entiers entre 2 bornes passées en paramètre.

Pour vérifier les résultats, toutes les valeurs affichées pourront être stockées dans une liste qui sera renvoyée à la fonction appelante

Exo 13 : le jeu « c'est plus, c'est moins »

Le fichier Python 'exo13-OCC_cestpluscestmoins.py' et celui d'explication 'exo13-OCC_cestpluscestmoins.md' , vous propose de deviner un nombre choisi par l'ordinateur dans un intervalle de 1 à 100 compris.

En fois, l'étude, l'analyse et la compréhension du fichier Python, veuillez répondre aux questions figurant dans le fichier d'extension `.md`.

Exo 14 : le jeu « pierre, feuille, ciseaux »

Le fichier Python 'exo14-OCC_PFC.py' et celui d'explication 'exo13-OCC_PFC.md' , vous propose de jouer à Pierre, Feuille, Ciseau contre l'ordinateur.

En fois, l'étude, l'analyse et la compréhension du fichier Python, veuillez modifier le programme fourni afin d'introduire dans le jeu, une nouvelle possibilité avec le 'Puits' avec les règles suivantes :

- Le puits bat la pierre et le ciseau
- Le puits est battu par la feuille

Questions subsidiaires :

- modifier le nombre de coups gagnants pour gagner la partie
- afficher en fin de partie tous les coups joués en gérant un historique

Exo 15 : le triangle de Pascal

Ecrire une fonction qui renvoie le triangle de Pascal sous la forme d'un dictionnaire où chaque clé correspond à un rang, et chaque valeur associée à la liste des coefficients

La fonction prendra le rang maximum pour lesquels les calculs seront à faire : par exemple pour un rang 5, la fonction renverra 6 clés pour des valeurs de 0 à 5 ;

Pour rappel les coefficients sont :

rang coefficients

0 → 1

1 → 1 1

2 → 1 2 1

3 → 1 3 3 1

etc …

Quelque soit le rang, les première et dernière valeurs sont à 1. Les valeurs intermédiaires se calculent à partir des valeurs du rang précédent. Par exemple, pour le rang 3 qui a 2 valeurs intermédiaires :

- La première valeur intermédiaire est égale à la somme de la valeur intermédiaire (même position) du rang précédent plus la valeur précédente de cette valeur du rang précédent : Ici 2 + 1 (valeur précédente en première position) donc 3.
- La seconde valeur est donc 1 + 2 (valeur précédente) qui vaut 3

Ce qui donne :

- 2 → *(1* + **[*2)* + 1]**
  ⇩ ⇩
- 3 → 1 *3* **3** 1

Le rang 4 a 3 valeurs intermédiaires, le rang 5 en a 4, etc ...

Exo 16 : introduction aux fonctions

En vous appuyant sur le travail fait sur l'exercice affichant le maximum entre 2 nombres :

1 - Ecrire une fonction en Python qui renvoie le maximum entre 2 variables passées en paramètres

2 – Ecrire une fonction en Python qui renvoie le maximum entre 3 variables passées en paramètres

3 – Ecrire une fonction qui renvoie le maximum entre plusieurs variables passées en paramètres

Exo 17 : structuration de données et fonctions

Ecrire une fonction `decimal_2_roman` qui transforme un nombre entier sous forme d'une string de chiffres romains. Pour rappel :

les unités sont : I, II, III, IV, V, VI, VII, VIII, IX
les dizaines sont : X, XX, XXX, XL, L, LX, LXX, LXXX, XC
les centaines sont : C, CC, CCC, CD, D, DC, DCC, DCCC, CM
les milliers sont : M, MM, MMM

Par exemple : 3486 s'écrit MMM CD LXXX VI

Voir https://fr.wikipedia.org/wiki/Numération_romaine

Exo 18 : étude et finalisation d'un script

A partir du fichier fourni – `**Model_superviseFolder.py**` – compléter les 2 fonctions suivantes :

- `snapshot_folder`
- `compare_folder`

La première fonction doit renvoyer une structure de données contenant les datas associées aux fichiers d'un dossier passé en paramètre, yc les sous-dossiers.

Ce sont ces 2 structures de données qui seront utilisées dans la fonction `compare_folder` et qui permettront d'indiquez quels sont :

- les fichiers ajoutés
- les fichiers supprimés
- les fichiers modifiés

Exo 19: analyse d'un texte, fonctions et valeurs retournées

Ecrire une fonction «**analyse_texte**» qui calcule et renvoie 3 valeurs :la chaîne la plus courte, la chaîne la plus longue et la longueur moyenne des chaîne de caractères d'un nombre variable de valeurs (tout type possible) passées en argument comme suit :

- **analyse_texte**('hello', 2.234, 't') → renvoie **'t', 'hello', 3.0**
- **analyse_texte**('ici', 'JE', 'assez', 13, 'salsifi', 'moutarde', -2.87, True) → renvoie **'JE', 'moutarde', 5.0**
- **analyse_texte**(-1) → **error - pas de chaine**
- **analyse_texte**() → **error**

Votre fonction devra traiter les erreurs avec le système de gestion d''erreur propre à Python. Attention, la fonction prend au moins un paramètre en entrée.

Exo 20: création d'une classe de base

Ecrire une classe 'Parallelo' qui prend en entrée les 3 paramètres suivants :

- un petit coté
- un grand coté
- un angle

et qui propose, en plus des méthodes de représentations internes, 2 méthodes publiques :

- « perimetre' qui calcule le périmètre du parallélogramme
- « surface' qui calcule la surface du parallélogramme

Vous devrez vérifier que les valeurs en entrée :

- sont toutes positives,
- que le petit coté est inférieur au grand coté
- que l'angle est compris entre 45 et 90 degrés

En cas de non respect de ces contraintes, l'objet ne pourra pas être créé.

Puis vous mettrez en place les méthodes internes nécessaires pour que :

- 2 objets puissent être comparées
- Une liste contenant `n' objets puissent être triée
- Un objet de cette classe puisse être ajouté à un set

En bonus, vous modifierez certaines méthodes de votre classe pour qu'un objet soit comparable avec un nombre flottant.

Exo 22: héritage d'une classe

A partir de la classe 'Parallelogramme', précédemment créée, vous créerez 2 classes filles 'Rectangle' et 'Losange' qui héritent de cette classe.

Vous vérifierez que les calculs sont toujours valides pour ces 2 classes. De plus, afin d'avoir des représentations internes cohérentes, vous redéfinirez les 2 méthodes de représentations des objets en Python.

Pour vérifiez toutes ces demandes, vous créerez 5 ou 6 objets avec au moins un exemplaire de chaque classe que vous ajouterez dans une liste et que vous afficherez à l'écran.

Vous vérifierez que ces objets sont toujours comparables et utilisables dans un ensemble.

Exo 24: héritage d'une classe, suite

A partir des classes définies à l'exercice 22, vous créerez une classe fille Carre.

Vous redéfinirez mes méthodes nécessaires afin de rester cohérents au moins au niveau des représentations internes.

Vous afficherez à l'écran la liste des classes dont hérite cette classe

Exo 26: mise en œuvre des `properties` dans une classe

Faire l'exercice 20. Puis positionner des 'property' pour toutes les attributs de la classe.

Vous vérifierez que vous avez autant de variables de classe de type 'property' que d'attributs en affichant la liste de ces variables.

Exo 30 : les notations en compréhension

Soit un dictionnaire m = {'a': 123, 'b': 456, 'c': 789}

Ecrire les 3 fonctions suivantes avec uniquement des notations en compréhension :

1) exclude(m, ('a', 'c')) # {'b': 456}
2) only(m, ('b', 'c')) # {'b': 456, 'c': 789}
3) values_at(m, ('a', 'b')) # [123, 456]

Exo 35 : introduction au module `collections`

Ecrire une fonction « **creer_index** » qui prend comme paramètre entrant une chaîne de caractère et qui renvoie un dictionnaire dont :

- les clés seront constituées de chaque mot distinct contenu dans la chaîne en entrée
- leur valeur respective, du nombre de fois ou le mot apparaît dans la chaîne en entrée.

dict_mot = **creer_index(**"ton tonton est ton ami et un ami de ton ami est mon ami"**)** →
renvoie **{'ami': 4, 'ton': 3, 'est': 2, 'de': 1, 'un': 1, 'mon': 1, 'et': 1, 'tonton': 1}**

Une évolution consisterait a limiter le nombre de clés du dictionnaire en fonction de la valeur associée, soit par exemple : seuls les mots qui apparaissent au moins 3 fois seront dans le dictionnaire : ce qui donnerait pour l'exemple ci-dessus :

dict_mot = creer_index("ton tonton est ton ami et un ami de ton ami est mon ami"**)** →
renvoie **{'ami': 4, 'ton': 3}**

Comment faire pour que cette restriction soit paramétrable ?

Exo 40 : les décorateurs

1 - sous forme de fonctions, écrire autant de décorateurs qu'il y a d'ingrédients et les appliquer à une fonction protéines de manière à afficher la composition d'un burger comme suit :

- PAIN
- ketchup
- salade
- fromage
- ma_protéine
- mourtarde
- PAIN

La fonction proteine est définie comme suit ;

```
def proteines() :
    print('poulet')
```

C'est donc cette fonction qui sera décorée.

2 – Produire le même burger sur une nouvelle fonction viande() avec un décorateur unique, appliqué autant de fois qu'il y aura d'ingrédients. Ce décorateur prendra en paramètre au moins le nom de l'ingrédient, voire d'autres paramètres que vous jugerez utile.

Exo 45: extraction de nombre depuis un itérable via une fonction

A partir d'un itérable (List, Tuple, Dict, Set, FrozenSet) passé en paramètre, extraire tous les éléments numériques contenus dans cet objet y compris dans ses sous-iterables qu'il contient. Renvoyer les éléments trouvés dans un tuple.

Pae exemple pour l = [12,'ddd', 1.24, True, (22, 2233), {'1':(5555, 4444),'deux': {tuple(), True,'False'}}]

renvoie (12, 1.24, True, 22, 2233, 5555, 4444, True)

PS : L'ordre n'a pas d'importance

Exo 46 : extraction de nombre depuis un itérable via un générateur

Reprendre l'énoncé précédent et transformer la fonction en générateur.

Exo 55 : les exceptions

La fonction python `exec(string)` lance l'interpréteur Python et évalue le contenu d'une chaîne de caractère passé en paramètre. Cf https://docs.python.org/3/library/functions.html#exec
Par exemple exec('10+2') donne 12. Cette valeur retournée peut être affectée à une variable.

Si le contenu lève une exception, celle-ci peut être interceptée de manière classique dans un bloc try/except/else

A partir du fichier python fourni – `**test_exception_vide.py**`, compléter les strings vides ("") du tuple **data_tests** par une ou plusieurs instructions Python qui vont déclencher l'exception.

Par exemple pour la première ligne de data_tests :

L'exception qui sera levée sera un `TypeError` pour la chaîne d'instruction " s = 'hello' ; s[0] = 'H' "

Exo 60 : spécialisation d'une liste à partir de `collections.UserList`

A partir de la classe collections.UserList, créer une nouvelle classe qui n'accepterait que des éléments numériques.
Tout ajout d'un élément non numérique provoquerait une levée d'exception.

Par exemple lnum = ListNum([True, 12, 6.9]) serait acceptée alors que ln2 = ListNum(['Hello', 4]) lèverait une exception

Vous aurez :

- dans un premier temps à chercher toutes les fonctions d'ajouts et/ou de modification d'éléments afin de redéfinir leur comportement initial.
- dans un deuxième temps, vous pourriez créer une classe plus générique qui permettrait de passer à la construction les types attendus pour cette UserList

Exo 65 : spécialisation d'une string à partir de `collections.UserString`

A partir de la classe collections.UserString, créer une nouvelle classe qui permettra pour le traitement des sous-chaine (slicing) de passer comme paramètre, en plus de l'existant (index, slice), un tuple qui pourra contenir une combinaison de slices, index et d'autres tuples

Par exemple :

s = **'hello les ami'**
s[1:4, 6, 10::-1] donnerait 'elllima' qui correspond à 'ell' joint à 'l' joint à 'ima'

Vous aurez donc à travailler sur la méthode __getitem__ .
Dans un deuxième temps, proposer une solution pour faire des affectations multiples en redéfinissant la méthode '__setitem__'

Exo 70 : les gestionnaires de contexte

1 - Ecrire un 'context manager' qui permet de mesurer le temps d'exécution d'une fonction

2 – Ecrire un 'context manager' qui permet de rediriger la sortie standard 'stdout' vers une autre sortie (un fichier, un pipe, un socket , etc …) pour toutes les instructions 'print' situées dans le bloc du Context Manager.