C2 - Python

C-COD-240

Jour 01

Apprendre Python

1.4

Jour 01

méthode de livraison: pythond01 sur Github

- La totalité de vos fichiers sources, à l'exception de tous les fichiers inutiles (binaires, fichiers temporaires, fichiers obj,...), doivent être inclus dans votre livraison.
- Les messages d'erreur doivent être écrits sur la sortie d'erreur, et le programme doit alors quitter avec le code d'erreur 84 (0 s'il n'y a pas d'erreur).

EXERCICE 01 (1PT)

Dossier à remettre :python_d01/ex_01/script.py

Créez un script avec le shebang adapté et les droits UNIX.
Il doit compter le nombre de caractères alphanumériques contenus dans les paramètres du programme. Le script doit afficher ce nombre sur la sortie standard.

Exemple:

```
Terminal
~/C-COD-240>./script.py 'Bonjour le monde !' '@_@' 'Geckos' 16
```

EXERCICE 02 (1PT)

Dossier à remettre :python_d01/ex_02/my_args_handler.py

Créez une fonction nommée « my_args_handler » qui prend plusieurs chaînes comme paramètre.
Cette fonction renvoie**un**chaîne qui est la concaténation de tous les arguments séparés par « , « .

Exemple:

```
Terminal
~/C-COD-240>python3
>>importer mon_gestionnaire_args
>>print(my_args_handler.my_args_handler("a", "b", "c", "poke","jour" "nuit")) a, b, c, poke, jour, nuit
```

EXERCICE 03 (1PT)

Dossier à remettre :python_d01/ex_03/double_return.py

Créez une fonction « double_return ».
Il prend un dictionnaire comme paramètre et renvoie deux listes.

Ces listes renvoyées doivent contenir des valeurs différentes.

Dans le premier, il y a la liste des clés et le second la liste des valeurs contenues dans le paramètre Dictionnaire.

Exemple:

```
Terminal
~/C-COD-240>python3
»>importer double_return
»>imprimer(double_return.double_return({“a”: 1, “b”: 2})) (['a',
'b'], [1, 2])
```

EXERCICE 04 (0PT)

Dossier à remettre :python_d01/ex_04/loader.py
python_d01/01_basics_04/required.py
Restriction:loader.py ne doit pas contenir « = »

Dans « required.py », créez une constante « REQUIRED » définie sur true.
Dans le chargeur, vous devez inclure et avoir accès à cette constante, quelle que soit la manière dont nous chargeons votre fichier « loader.py »

EXERCICE 05 (1PT)

Dossier à remettre :python_d01/ex_05/show_methods.py

Créez une fonction « show_methods ».
Il prend un paramètre d'objet et affiche toutes ses méthodes.

Exemple:

Terminal

```
~/C-COD-240>python3
»>à partir de show_methods importer
show_methods »>show_methods(“un objet chaîne”)
['ajouter', 'classe', 'contient', 'delattr', 'dir', 'doc',
'équ.', 'format', 'ge', 'obtenir un attribut', 'obtenir l'article', 'obtenir de
nouvelles ressources', 'gt', 'hacher', 'init', 'init_subclass', 'itérer', 'le', '
longueur', 'lt', 'mod', 'mul', 'ne', 'nouveau', 'réduire', 'réduire_ex',
'représentant', 'rmod', 'rmul', 'setattr', 'taille de', 'str', 'crochet de sous-classe',
'capitaliser', 'casefold', 'centrer', 'compter', 'encoder', 'endswith', 'développer les onglets',
'trouver', 'format', 'format_map', 'index',
'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric',
'isprintable', 'isspace', 'istitle',
'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind',
'rindex', 'rjust',
'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'traduire',

'supérieur', 'zfill']
```

EXERCICE 06 (1PT)

Dossier à remettre :python_d01/ex_06/mon_show_platypus.py

Créez une fonction nommée « my_show_platypus » qui prend un nombre comme paramètre et qui affiche « Platypus » autant de fois qu'indiqué par le paramètre.
Chaque représentation de « Platypus » sera suivie d'une nouvelle ligne.

Exemple:

```
Terminal
~/C-COD-240>python3
»>importer mon_show_platypus
»>mon_spectacle_platypus.mon_spectacle_platypus(5)
Ornithorynque
Ornithorynque
Ornithorynque
Ornithorynque
Ornithorynque
```

EXERCICE 07 (1PT)

Dossier à remettre :python_d01/ex_07/display_all.py

Créez une fonction « display_all ».
Il prend un dictionnaire comme paramètres et affiche tous les éléments du dictionnaire, formatés comme suit : « (classe : valeur) » suivi d'une nouvelle ligne

- « [[classe]" : doit être la classe de l'élément.
- « [[valeur] : sa valeur convertie en chaîne.

Exemple:

```
Terminal
~/C-COD-240>python3
»>émissions d'importation
»>shows.shows({"test": "bonjour", "t": 43, "pi": 3.14}) (str:
bonjour)
(int: 43)
(flotteur : 3,14)
```

EXERCICE 08 (1PT)

Dossier à remettre :python_d01/ex_08/display_all.py

Créez une fonction « display_all ».
Il prend un dictionnaire comme paramètre et affiche chaque élément du dictionnaire formaté comme suit :
« clé->(classe : valeur) » suivi d'une nouvelle ligne

Exemple:

```
Terminal
~/C-COD-240>python3
»>émissions d'importation
»>shows.shows({"test": "bonjour", "t": 43, "pi": 3.14}) test->(str:
bonjour)
t->(int: 43)
pi->(float: 3.14)
```

EXERCICE 09 (2PTS)

Dossier à remettre :python_d01/ex_09/display_all.py

Créez une fonction « display_all ».

Il prend un dictionnaire comme paramètre et affiche chaque élément du dictionnaire formaté comme suit : « id->(classe : valeur) » suivi d'une nouvelle ligne

« id » est l'index, de 0 à la longueur du dictionnaire.

Remarque : « = » est interdit uniquement pour les assignations, et non pour les paramètres par défaut. Cette règle s'applique à tous les exercices suivants. Cela signifie que vous ne pouvez pas créer ni modifier les valeurs des variables/arguments.

Exemple:

```
Terminal
~/C-COD-240>python3
»>émissions d'importation
»>shows.shows({"test": "bonjour", "t": 43, "pi": 3.14}) 0->(str:
bonjour)
1->(int: 43)
2->(float: 3.14)
```

EXERCICE 10 (1PT)

Dossier à remettre :python_d01/ex_10/increment.py **Obligatoire:**Utiliser les capacités de compréhension de liste/dictée de Python

Créez une fonction « incrément » qui prend une liste en paramètre et renvoie la même liste avec tous les éléments entiers augmentés de un :

Exemple:

```
Terminal
~/C-COD-240>python3
»>incrément d'importation
»>incrément.incrément([3, 4, 9, 'a', “qvb' ', {}]) => [4, 5, 10,
'a', “qvb' ', {}]
```

EXERCICE 11 (2PT)

Dossier à remettre :python_d01/ex_11/shows.py **Obligatoire:**Utiliser les capacités de compréhension de liste/dictée de Python

Créez une fonction « shows ». Elle prend un dictionnaire en paramètre.

Pour chaque élément du dictionnaire, il doit le convertir en une chaîne suivant le format : « ([clé]) -> ([classe] : [valeur]) ».

La fonction imprimera une liste avec chaque valeur convertie.

Exemple:

```
Terminal
~/C-COD-240>python3
»émissions d'importation
»>shows.shows({“test”: “hello”, “t”: 43, “pi”: 3.14}) (t)->(int: 43)

(pi)->(float: 3,14)
(test)->(str: bonjour)
```

EXERCICE 12 (2PTS)

Dossier à remettre :python_d01/ex_12/count_words.py **Obligatoire:**Utiliser les capacités de compréhension de liste/dictée de Python

Créez une fonction « my_count_words » qui prend une liste en paramètre et renvoie un dictionnaire.

La liste passée en argument est remplie de chaînes.

Le dictionnaire renvoyé doit être formaté pour respecter les règles suivantes :

- Chaque clé est une chaîne.
- Chaque valeur est un entier.
- La valeur associée est le nombre d'occurrences de la clé dans le paramètre du dictionnaire.

EXERCICE 13 (3PTS)

Dossier à remettre :python_d01/ex_13/sum_sub.py
Obligatoire: Utiliser la réduction de functools

Créez un script python qui ajoutera ou soustraira les valeurs passées en arguments et en fonction de 3 règles simples :

- S'il n'y a pas d'arguments, affichez 0.
- Sinon, la première valeur à utiliser est le premier argument.

- Pour chaque argument suivant, si la valeur actuelle est impaire, ajoutez l'argument suivant à la valeur actuelle, sinon soustrayez-le.

EXERCICE 14 (3PTS)

Dossier à remettre :python_d01/ex_14/regexp.py

Créez une constante globale EMAIL. Il doit s'agir d'une instance d'expression régulière pour capturer un e-mail.
Un e-mail doit respecter la RFC 5322.
Mais votre exercice sera plus facile.

Vous n'implémenterez pas une RFC complète.
Respectez la grammaire spécifiée :

- EMAIL = [UTILISATEUR@DOMAINE]
- UTILISATEUR = [tout(1, 1023)]
- DOMAINE = [SOUS-DOMAINE ['.' SOUS-DOMAINE]*] & [any(1,253)]
- SOUS-DOMAINE = [alphaalpha | num]

- * =0, 1 ou n fois
- (a,b) = entre a et b fois
- alpha = tout caractère alphabétique
- num = tout caractère numérique (0-9)
- any = tout caractère non nul
- =une union logique (OU LOGIQUE)
- & =une intersection logique (LOGIQUE ET)

Le domaine complet et l'utilisateur doivent être capturés et nommés.

Valide:
utilisateur@domaine.domaine.domaine.domaine

Non valide:

utilisateur@domaine.domaine.domaine.domaine.domaine.domaine.domaine.domaine.domaine.domaine.domaine.domaine.domaine.domaine