Traduit de Anglais vers Français - www.onlinedoctranslator.com

C2 - Python

C-COD-240

Jour 02

Premier projet

1.4

Jour 02

méthode de livraison: pythond02 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).

C*ding Academy

EXERCICE 01 (2PTS)

Dossier à remettre :python_d02/time_01/display_time.py

Créez une fonction « display_time ». Elle prend comme argument une date au format epoch (en secondes depuis le 01/01/1970 00:00).

Il affiche le paramètre en GMT en suivant l'exemple :

```
Terminal
~/C-COD-240>python3
»>à partir de display_time importer display_time
# Le 27/04/2017 à 17h10 »>
affichage_heure(1493313015)
17:10:15, jeudi, 4* jour de la 17* semaine de 2017, avril
# Le 24/02/1993 à 8h »>
affichage_heure(730540800)
08:00:00, mercredi, 3* jour de la 08* semaine de 1993, février
```

Convertissez les dates en ligne avecce lien

EXERCICE 02 (3PTS)

Dossier à remettre :python_d02/classes_modules_01/email.py

Créez une classe « Email ».
Il possède 7 attributs suivant les règles ci-dessous :

- _corps : paramètre du constructeur
- _objet : Paramètre du constructeur
- _de : paramètre du constructeur
- _à : paramètre du constructeur
- _created_at : défini lors de l'initialisation à l'heure actuelle
- _updated_at : défini lors de l'initialisation à l'heure actuelle
- _sent_at : défini lors de l'initialisation à None

Voir aussi Arguments de mots-clés et Paramètres positionnels.

EXERCICE 03 (3PTS)

Dossier à remettre :python_d02/classes_modules_02/email.py,
python_d02/classes_modules_02/sendable.py,
python_d02/classes_modules_02/sms.py

Copiez le code précédent de la commande Email, mais renommez la classe « Sendable » et le fichier « sendable.py ».
Créez maintenant deux fichiers contenant les classes « Email » et « Sms ».

Ils héritent de Sendable.

Le courrier électronique et les SMS utilisent le constructeur parent.
L'e-mail prend les mêmes paramètres qu'avant, et les SMS ne prennent que les paramètres « corps », « de » et « à ».

De plus, dans Sendable, ajoutez la méthode « send ». Elle définit l'attribut « sent_at » à l'heure actuelle, mais une seule fois.

L'appel de cette méthode deux fois doit déclencher une exception « DataAlreadySent » sans message.

Remarque : assurez-vous d'importer les modules requis avant d'utiliser les classes

EXERCICE 04 (3PTS)

Dossier à remettre :python_d02/classes_modules_03/email.py,
python_d02/classes_modules_03/private.py,
python_d02/classes_modules_03/sendable.py,
python_d02/classes_modules_03/sms.py

Copiez le code précédent.
Désormais, chaque appel à « sent » enregistrera les métadonnées de l'objet Sendable dans un historique, global pour tous les objets Sendable.

Ces métadonnées sont « de », « à » et « envoyé à ».
Pour effectuer cette action, créez un nouvel attribut de classe dans « Sendable ». Il
doit s'agir d'un dictionnaire vide.
Lorsque « envoyé » est appelé avec succès, il doit enregistrer les informations.

La structure attendue est :

> {de => { à => [sent_at, sent_at, sent_at, . . .], à => [sent_at, sent_at, . . .] . . . }}

Ensuite, ajoutez une méthode de classe « history » pour le lire.

Créez ensuite une classe « Private » comme « Email ». Cependant, lorsque la méthode « sent » est appelée, vous ne devez déclencher aucune action d'espionnage, ni même modifier la variable sent_at.

Mais une erreur doit toujours se produire si le message est envoyé deux fois. Soyez malin.

EXERCICE 05 (3PTS)

Dossier à remettre :python_d02/class_modules_04/email.py,
python_d02/class_modules_04/sendable.py,
python_d02/class_modules_04/sms.py,
python_d02/class_modules_04/private.py

Copiez le code précédent.

Créer une nouvelle méthode**crypter**qui prend un paramètre optionnel**tourner**. Si la
rotation n'est pas spécifiée, vous devez utiliser la valeur par défaut 13.
Lorsque cette méthode est appelée, vous devez appliquer une opération ROT-13 sur le corps (ou ROT-N si la rotation est spécifiée).

La méthode renvoie le texte chiffré.
Ajouter une méthode**décrypter**pour faire l'opération inverse (même paramètre, renvoie le texte décrypté).

EXERCICE 06 (3PTS)

Dossier à remettre :python_d02/classes_modules_05/email.py,
python_d02/classes_modules_05/private.py,
python_d02/classes_modules_05/sendable.py,
python_d02/classes_modules_05/sms.py, python_d02/
classes_modules_05/sendable_box.py

Copiez le code précédent.

Ajoutez une classe SendableBox.
Il dispose d'un attribut de classe « all » pour stocker chaque SendableBox créée.
Chaque fois que vous créez une nouvelle SendableBox (avec un argument, son adresse), vous devez créer une nouvelle boucle infinie dans un thread.
Toutes les secondes, il vérifie s'il reçoit un nouveau message.
Si vrai, affichez « Nouveau message de {m.from} » sur la première ligne, puis sur une nouvelle ligne l'objet ou le corps si le premier est Aucun.
Il affiche un message maximum par seconde.

SendableBox possède une fonction de classe « recv », prenant comme argument un Sendable et devant le distribuer.

Liez Sendable avec SendableBox lorsque send est appelé, déclenchez recv

Ajoutez ensuite un 2ème paramètre « max_box » au constructeur SendableBox.
Lorsque le nombre de messages reçus par la SendableBox est égal à max_box, la boucle dans le thread doit être interrompue
Enfin, ajoutez un troisième paramètre « sleep_duration ». Au lieu d'attendre une seconde entre chaque vérification, attendez seulement « sleep_duration » secondes.

EXERCICE 07 (3PTS)

Dossier à remettre :python_d02/classes_modules_06/contact.py

Dans cet exercice, vous découvrirez les capacités d'héritage multiple de Python.
Créez 3 classes Addressable, Nameable et HasFriend.

- Addressable doit définir un attribut « adresse »
- Nameable doit définir un attribut « nom »
- HasFriend doit définir un attribut « friends » qui renvoie le tableau des amis de la classe qui les inclut.

Si les attributs amis n'existent pas, il faut les créer.
Créez une classe Contact qui héritera des 3 classes précitées.