R504-TP2

TDD versionné en Python

Merci de lire l'entièreté du document avant de commencer.

Introduction : Commencer par créer un dossier de travail que vous allez initier comme un dossier GIT. Vous le lierez ensuite à un dépôt distant sur Github.
Vous vous assurez de créer des commits tout au long de votre développement et à chaque étape.

Rappel sur le TDD :

- Création d'un test qui FAIL
- Création du code qui le fait SUCCESS
- Refactorisation de la solution en gardant SUCCESS
- Ecriture d'un nouveau test qui FAIL….etc, etc jusqu'à La Solution à livrer

Il est très important de respecter cela car c'est sur cela que vous allez être corrigé.
En guise de compte-rendu tout ce que vous aurez à faire c'est nous inviter sur votre dépôt distant en tant que collaborateur pour que l'on puisse accéder à l'entièreté des informations de vos commits.
Cela nous permettra de voir avec précision et dans un ordre chronologique comment vous êtes arrivé à la solution.

Question 1 : *FizzBuzz (environ 1 heure 3O)*

A. *Création d'une première solution :*

Créer une méthode affiche(), qui ne prend rien en paramètre et qui affiche de manière concaténée, les numéros de 1 à 100.
Elle doit remplacer les multiples de 3 par "Fizz"; les multiples de 5 par "Buzz" et les multiples de 3 et de 5 (de 15) par "FrisBee".

Exemple d'exécution :

```
affiche()
12Fizz4BuzzFizz78FizzBuzz11Fizz1314FrisBee1617Fizz…..9798FizzBuzz
```

B. *Première évolution du code :*

On veut maintenant que la méthode prenne un paramètre n de type nombre.
Elle retournera les chiffres de 1 à n selon les mêmes règles.
Redéfinissez vos tests initiaux pour les faire passer.
Continuez de nouveau test pour arriver à la solution.

Exemple d'exécution :

```
affiche(15)
12Fizz4BuzzFizz78FizzBuzz11Fizz1314FrisBee
```

C. *Deuxième évolution du code :*

La fonction affiche() prend maintenant deux nombres en entrée n1 et n2.
Elle affiche selon les mêmes règles les chiffres concaténés de n1 à n2 inclus.

Exemple d'exécution :

```
affiche(5, 10)
BuzzFizz78FizzBuzz

affiche(10, 16)
Buzz11Fizz1314FrisBee16
```

Question 2 : *Cryptage (environ 1 heure 30)*

A. *Création de la première solution :*

Créer avec une méthodologie TDD, une fonction crypt(message) qui prend en paramètre un string et qui renvoie ce message de manière cryptée.
On veut que chaque lettre du message soit remplacée par la suivante dans la table ASCII.

Astuce :

Vous pouvez stocker la table à ski dans une variable en utilisant l'objet String de la manière suivante :

```
import string

caracteres = string.ascii_letters + string.ascii_punctuation + string.ascii_digits + " "
```

N’oubliez pas d’y concaténer un espace comme dans l’exemple que je viens de vous donner.

B. *Première évolution du code :*

On veut maintenant que notre méthode crypt(message, pas) prenne un deuxième argument de type entier, compris entre 1 et 9 et qui décale de ce nombre le cryptage dans la table ASCII.

Elle retourne le message crypté avec le pas concaténé à la fin du message.

C. *Deuxième évolution du code :*

On veut maintenant et toujours en TDD, créer une seconde méthode decrypt(message), pour pouvoir déchiffrer les messages retournés par crypt.