TD4-Mini-projets-Niveau1

October 6, 2021

1 TD 4 : Mini projets de niveau 1

1.1 Une introduction

Pour être complet, un projet informatique devrait présenter un pseudo-code. Un pseudo-code est une présentation d'un algorithme le plus proche du langage comme le français. Nous ne l'imposons pas pour ce TD.

1.2 Les 8 projets

Les huit projets ici proposés vont permettre de tester vos capacités algorithmiques. + 1. Pésentation : Il est essentiel que vous présentiez votre code en français en expliquant brièvement son fonctionnement. + 2. Commentaires Python : n'oubliez pas de commenter les étapes de votre code. + 3. Les tests : le jeu de tests que vous proposerez afin de valider votre programme fait partie pour la moitié de la note attribuée à l'exercice. Vous devez commenter vos tests et exécuter vos fonctions.

1.3 Conseils

- Vous pouvez développer vos projets sur un éditeur python comme Spyder (compris dans Anaconda) ou repl.it puis recopier le code sur votre notebooks.
- Commentez le code, exécuter le code, testez le code, commentez le test, retestez le code, commentez le test, retestez le code, commentez le test, retestez le code, commentez le test ….

1.4 Notations : devoir sur 48 points

- Chaque exercice est noté sur 6 points.
- 3 points pour le code + 3 points pour les tests.
- Même si chaque exercice est noté sur 6 points, ils ne sont pas tous de la même difficulté, certains sont très simples et d'autres plus ardus.

2 Projet I (Prologin 2019) : Arnaque aérienne - Niveau 1 (6 points)

2.1 Énoncé

Comme le disait le grand-père de Joseph Marchand, « Le plus beau voyage est celui qu'on n'a pas encore fait ». New York était dans la tête de Joseph depuis plusieurs années maintenant, et il a décidé aujourd'hui d'acheter son billet d'avion.

Quelques secondes avant de cliquer sur le bouton « Acheter ! », son amie Haruhi lui envoie une liste de prix qu'elle a trouvés sur Internet. Joseph est curieux de voir si ces derniers sont moins chers que le billet qu'il s'apprêtait à acheter.

2.2 Entrée

- Le prix initial du billet de Joseph.
- Une liste contenant les N prix trouvés par Haruhi..

2.3 Sortie

- Si Haruhi a trouvé au moins 3 prix strictement moins chers que celui de Joseph, affichez « ARNAQUE ! » pour l'avertir.
- Sinon « Ok bon voyage, bisous, n'oublie pas de m'envoyer des photos ! ».

[ ]:
```
#Exemples d'entrée/sortie

# Dans la console Python
prixInit=570
listeHaruhi= [495, 1200 ,540 ,450]
arnaqueAerienne(prixInit,listeHaruhi)
"ARNAQUE"
```

2.4 Commentaire

Exactement 3 billets sont strictement moins chers que celui choisi par Joseph : 495, 540, et 450. Ça sent l'arnaque..

[ ]:
```
# Dans la console Python
prixInit=820
listeHaruhi= [580, 2000, 970, 1050, 820]
arnaqueAerienne(prixInit,listeHaruhi)
"Ok bon voyage, bisous, n'oublie pas de m'envoyer des photos !"
```

2.5 Commentaire

Seul le billet à 580€ est strictement inférieur au billet de Joseph à 820€.

2.6 Rendu

- Un code python dans le bon format.
- Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.

[1]:
```
#Mettre le code python ici: (3 points)
```

Présentation de votre code

[ ]:
```
#TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS EST EVALUES (3 points)
```

+++
Projet II (Prologin 2011) : Les « 101 » Dalmatiens - Partie I – Niveau 1 ## Énoncé

Joseph Marchand veut faire une farce à son fidèle dalmatien Scooby-Naire : il souhaite accrocher dans sa niche une photo de ce bel animal, en inversant au préalable les couleurs blanc et noir.

On vous donne un tableau de bits représentant une image en noir et blanc, vous devez implémenter une fonction qui calcule le négatif de cette image.

2.7 Entrée

Le tableau de bits (N x M) fait avec une liste de liste, par exemple pour un tableau 3x5:

[ ]:
```
#dans l'éditeur PYTHON
exemple=[[0,0,0,1,0],[1,1,1,1,1],[1,0,1,0,1]]
#Comme on parle d'un tableau, on l'écrira plutôt
exemple=[[0,0,0,1,0],
         [1,1,1,1,1],
         [1,0,1,0,1]]
#Remarquez que les deux exemples precedents sont EXACTEMENT les mêmes. Seul␣
 ↪l'affichage est different
```

2.8 Sortie

Le négatif de l'image (N x M) fait avec une liste de liste.

2.9 Exemples d'entrée/sortie

[ ]:
```
#Exemples d'entrée/sortie
#dans l'éditeur PYTHON
monChien=[[0,0,0],
         [1,0,1],
         [0,1,0]]

dalmatienI(monChien)
[[1,1,1], [0,1,0],[1,0,1]]
```

2.10 Rendu

- Un code python dans le bon format.
- Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
- Un jeu de tests aléatoires.

[ ]:
```
#Mettre le code python ici:
```

[ ]:
```
##TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS SONT EVALUES
```

Commentaires

++
Projet III (PROLOGIN 2010) : ## Énoncé

Une séquence d'ADN sera une suite finie constituée de lettres dans l'ensemble {A, T, G, C}. On vous donne en entrée une séquence d'ADN de longueur N. Écrivez une fonction qui renvoie le nucléotide (la lettre) le plus présent. Si c'est le cas de plusieurs, renvoyez celui qui vient en premier dans l'ordre alphabétique.

2.11 Entrée

Une chaîne de caractères, la séquence d'ADN de longueur N.

2.12 Sortie

Le nucléotide le plus fréquent dans la séquence d'ADN. ## Exemples d'entrée/sortie

[ ]:
```
#Exemples d'entrée/sortie
# Dans la console Python
print(freqADNPlus("ATTGCCATATCC"))
C
print(freqADNPlus("AAAACCCGGGTTT"))
A
```

2.13 Rendu

- Un code python dans le bon format.
- Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
- Un jeu de tests aléatoires.

[ ]:
```
#Mettre le code python ici:
```

[ ]:
```
#TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS SONT EVALUES
```

Commentaires

++
Projet IV (PROLOGIN 2009) : Timestamps – Niveau 1 ## Énoncé

En épreuve machine de demi-finale, vous soumettrez des solutions pour divers problèmes. On vous donne six variables entières représentant trois heures de soumission (données sous la forme de deux entiers, l'un pour les heures et l'autre pour les minutes). Écrivez une fonction qui détermine l'heure de la première soumission. Les trois heures données à votre programme seront toujours différentes. ## Entrée Une liste de trois couples d'entier, le premier étant l'heure de soumission et le second, les minutes. Sur chaque ligne, les deux entiers seront séparés par un espace.

2.14 Sortie

Renvoyez 1, 2, ou 3, si la première, la deuxième, ou la troisième heure est la première soumission.

2.15 Exemples d'entrée/sortie

[ ]:
```python
#Exemples d'entrée/sortie
# Dans la console Python
soumission=[(10,48),(9,3),(9,39)]
timeStamps(soumission)
2
soumission=[(7,58),(20,26),(9,29)]
timeStamps(soumission)
1
```

2.16 Rendu

- Un code python dans le bon format.
- Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
- Un jeu de tests aléatoires.

[ ]:
```python
#Mettre le code python ici:
```

[ ]:
```python
#TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS SONT EVALUES
```

++
Projet V (Prologin 2008) : Les assiettes – Niveau 1 ## Énoncé

Joseph Marchand est un homme très ordonné et il souhaite ranger ses assiettes en trois piles égales. Écrivez une fonction qui prend en argument trois entiers représentant respectivement la hauteur des trois piles actuelles et qui retourne 1 s'il est possible d'obtenir trois piles égales, 0 sinon. ## Entrée

L'entrée compte 3 entiers, N, M et P, les hauteurs des trois piles d'assiettes. ## Sortie

La sortie contiendra True s'il est possible de rééquilibrer les piles d'assiettes, False sinon. Contraintes d'exécution ## Exemples d'entrée/sortie

[ ]:
```python
# Dans la console Python
N,M,P=1,2,3
lesAssiettes(N,M,P)
True
N,M,P=0,0,2
lesAssiettes(N,M,P)
False
```

2.17 Rendu

- Un code python dans le bon format.
- Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
- Un jeu de tests aléatoires.

[ ]:
```python
#Mettre le code python ici:
```

[ ]:
```python
#TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS SONT EVALUES
```

++
Projet VI (Prologin 2007) : Bissextile – Niveau 1 ## Énoncé

Déterminer si une année est bissextile.

Écrire une fonction qui prend une année (un entier) en argument et retourne True si elle est bissextile, False sinon. Attention une année n'est pas bissextile tout les 4 ans, c'est plus délicat.

2.18 Entrée

L'entrée ne contient qu'un seul entier: l'année>0.

2.19 Sortie

True si l'année est bissextile, False sinon.

2.20 Exemples d'entrée/sortie

[ ]:
```python
# Dans la console Python
print(is_bissextile(2000))
True
print(is_bissextile(2042))
False
print(is_bissextile(1800))
False
print(is_bissextile(2008))
```

2.21 Rendu

- Un code python dans le bon format.
- Un jeu de tests en fin de code contenant entre autre les trois exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
- Un jeu de tests aléatoires.

[ ]:
```python
#Mettre le code python ici:
```

[ ]:
```python
#TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS SONT EVALUES
```

++
Projet VII (Prologin 2006) : Parité - Niveau 1

2.22 Enoncé

Écrire une fonction qui teste si les nombres du tableau donné en argument sont alternativement pairs et impairs. La fonction affichera True si c'est le cas, False sinon. ## Entrée Une liste de N entiers.

2.23 Sortie

Vous devez renvoyer une booléen soit True, soit False.

2.24 Exemples d'entrée/sortie

[ ]:
```
# Dans la console Python
maListe1=[4, 5, 8, 3, 2, 5]
maListe2=[4, 8, 8, 8, 2, 5, 7, 2]
maListe3=[1,2,3,4, 7, 2]
print(parite(maListe1))
True
print(parite(maListe2))
False
print(parite(maListe3))
True
```

2.25 Rendu

- Un code python dans le bon format.
- Un jeu de tests en fin de code contenant entre autre les deux exemples précédents. Ne vous limitez surtout pas aux exemples précédents.
- Un jeu de tests aléatoires.

[ ]:
```
#Mettre le code python ici:
```

[ ]:
```
#TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS SONT EVALUES
```

++
Projet VIII : Code de César (chiffrements) - Niveau 1 ## Énoncé Écrire le code permettant de chiffrer un message. La méthode cryptographique utilisée est un décalage, type cryptage de César. Le code de Cesar consiste à décaler chaque lettre d'un mot d'un même valeur dans l'ordre alphabétique. Par exemple: "zoologie" devient "appmphjf" avec un decalage positif d'une lettre. "zoologie" devient "bqqnqikg" avec un decalage positifde deux lettres. "argon" devient "zqfnm" avec un decalage négatif d'une lettre.

La valeur de décalage est appelée clef de chiffrement.

On utilisera seulement des lettres minuscules sans accent (ASCII) ## Entrée : Une chaine de caractéres (str) en minuscule sans accent ## Sortie : Une chaine de caractéres (str) en minuscule sans accent ## Exemples d'entrée/sortie

[1]:
```
print(chiffreCesar("zoologie",1))
'appmphjf'
print(chiffreCesar("zoologie",2))
'bqqnqikg'
print(chiffreCesar("argon",-1))
'zqfnm'
print(chiffreCesar("beaute",26))
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-1-6b138c9e1ed4> in <module>
----> 1 print(chiffreCesar("zoologie",1))
      2 'appmphjf'
      3 print(chiffreCesar("zoologie",2))
      4 'bqqnqikg'
      5 print(chiffreCesar("argon",-1))


NameError: name 'chiffreCesar' is not defined
```

[ ]:
```python
#Le code python ici
def chiffreCesar(...):
    """
    IN :
    OUT:...
    """
    ...
```

[ ]:
```python
##TESTS , ATTENTION LA QUALITE ET LE NOMBRE DE TESTS SONT EVALUES
```