UniLaSalle Amiens I3-RIOC-FISA	CM/TD2 - 2h Python Alban MAZEROLLES

Les exercices sont totalement indépendants et peuvent être fait dans l'ordre de votre choix.

Veuillez-vous munir de vos PC afin de tester les bouts de code.

Le vocabulaire à connaître pour le DS est indiqué d'un *.

1 COPIE DE LISTE – LES PIEGES A EVITER

La copie d'une liste par égal peut faire penser que la copie et son original sont deux éléments dissociés. Ce n'est hélas pas le cas.

On exécute le code suivant :

```
a=[1,2,3]
b=a
b.append(4)
print("a: ", a)
print("b: ", b)
```

On obtient le résultat suivant :

```
('a: ', [1, 2, 3, 4])
('b: ', [1, 2, 3, 4])
```

Q1. Est-ce conforme à votre attente ? Quel nom donne-t-on habituellement à ce type de copie ?

Le comportement observé est un comportement typique **en python** de tous les objets ***modifiables***. On parle de ***mutables****.

Q2. Maintenant, on remplace `b=a` par `b=a[:]`. Qu'est-ce que cela change ?

Maintenant, refaisons cette étude avec un cas un peu plus complexe (listes imbriquées* – nested lists) :

```
a=[1,2,[4,5,6],3]
b=a
b.append (4)
b[2].append (7)
print ("a: ", a)
print ("b: ", b)
```

Q3. Recommencer les questions Q1 et Q2.

Q4. Essayer les instructions suivantes et concluez sur quand utiliser quoi.

```
import copy
a=[1,2,[4,5,6],3]
b=copy.deepcopy(a)
b.append (4)
b[2].append (7)
print ("a: ", a)
print ("b: ", b)
```

Q5. Rédigez-vous un pense bête (idéalement un vademecum) à ce sujet. Si vous avez besoin de conseils en termes d'outil, n'hésitez pas à demander.

2 PRECEDENTE SUBTILITE AVEC DES FONCTIONS…

Tout d'abord, découvrons une nouvelle syntaxe (ce n'est pas la plus utilisée, mais c'est toujours bon de l'avoir vu au moins une fois) :

```
def multipleByTwo(maliste: list):
    return list(map(lambda x: 2*x, maliste))

lst = [1, 2, 3, 9, 8, 7]
print(multipleByTwo(lst))
```

Q1. Expliquer précisément ce que fait la fonction ***map****.

Q2. Qu'est une fonction ***lambda**** ? A quoi sert-elle ?

Q3. Remplacer la ligne (avec le `return`) en utilisant une liste de compréhension pour aboutir au même résultat.

Q4. On rajoute maintenant à la toute fin du programme une nouvelle fois la ligne `print(multipleByTwo(lst))`. SANS TESTER, quel va être le résultat que l'on va obtenir. Faites MAINTENANT le test. Est-ce conforme à vos attentes ?

Passons à un autre code relativement similaire. Ici, la fonction propose une valeur par défaut (qui est une liste vide) :

```
def mettreaubout(number, number_list=[]):
  number_list.append(number)
  print(number_list)
  return number_list

mettreaubout(5)
mettreaubout(7)
```

Q5. SANS TESTER, qu'espérez-vous avoir comme résultat ?

Q6. TESTER et vérifiez si cela est conforme à vos attentes ?

Observez ce qui se passe avec cette légère modification :

```
def append(number, number_list=None):
  if number_list is None:
    number_list = []
  number_list.append(number)
  print(number_list)
  return number_list

append(5)
append(7)
```

Q7. * Concluez. SOIGNEZ BIEN cette question, c'est évidemment la question la plus importante de l'exercice.

3 ANALYSER ET COMPRENDRE LE PROGRAMME DE QUELQU'UN D'AUTRE

Voici le programme : (que vous retrouverez sur le Moodle sous l'intitulé : exo3.py)

```
#encoding:utf-8
from pick import pick
import pandas as pd
from statistics import mean

read_file = pd.read_excel (r'Classeur_notes_exemple.xlsx', sheet_name='Feuil1')
read_file.to_csv (r'Classeur_notes_exemple.csv', index = None, header=True)

df = pd.read_csv(r'Classeur_notes_exemple.csv')

notes_brutes = df['Note brute'].values.tolist()

min_mark = min(notes_brutes)
max_mark = max(notes_brutes)
delta_mark = max_mark - min_mark

### Atteinte d'une moyenne précise
wanted_avg_mark = 9.0

avg_mark = mean(notes_brutes)
diff_avg = wanted_avg_mark - avg_mark

notes = df['Note brute'].apply(lambda x: x + diff_avg)
if notes.max() > 20:
    notes = notes.apply(lambda x: x - (notes.max()-20) )

    y = 0
    delta = 1
    dir = "up"
    y_max = notes.max()
    y_min = notes.min()
    while True:
```

```
        _notes = notes.apply(lambda x: round((y_max - (y_min + y))/delta_mark*(x-max_mark)
+ 20, 2) )
        actual_diff_avg = wanted_avg_mark - _notes.mean()
        if actual_diff_avg > 0.001:
            if dir == "down":
                delta /= 2
            y += delta
            dir = "up"
        elif actual_diff_avg < -0.001:
            if dir == "up":
                delta /= 2
            y -= delta
            dir = "down"
        else:
            notes = _notes
            break

print(notes.apply(lambda x: round(x,2)))

print(notes.mean())

df["Note recalibrée"] = notes

print(df)

df = df.reset_index(drop=True)
try:
    with pd.ExcelWriter(r'Classeur_notes_exemple.xlsx', engine="openpyxl", mode='a') as
writer:
        df.to_excel(writer, sheet_name='Feuil2')
except (PermissionError):
    print("Veuillez fermer le fichier svp.")
except (ValueError):
    print("L'onglet demandé existe déjà.")
```

Q1. Quel objectif permet d'atteindre ce programme ;

Q2. Détaillez l'ensemble des étapes du programme qui permet d'atteindre cet objectif.

Q3. *Justifiez l'utilisation des blocs `try : … except : …`

Q4. Expliquez **chacune** des lignes utilisant le module `pandas`.

Q5. A quoi sert cette variable `dir` que l'on voit à plusieurs reprises dans le programme.

Q6. *A quoi servent les fonctions `lambda`. Sont-elles obligatoires ? Quel confort apportent-elles ?

Q7. Ce code pourrait être bien plus concis et élégant. C'est le moment de montrer vos talents de programmeur !