Université de Lille 1 SCIENCES ET TECHNOLOGIES

ufr d'IEEA
Formations en
Informatique de
Lille 1

L1 MASS/SESI/PEIP - 2014–2015

Algorithmes et Programmation 1

Devoir surveillé 1

Samedi 28 février 2015 - durée 2h00 - documents non autorisés

Veuillez indiquer le numéro de votre groupe de TD sur la copie qu'il est inutile de rendre anonyme.
Ce sujet contient 4 exercices indépendants.
Toute fonction que vous réaliserez devra être documentée et les contraintes d'utilisation précisées.

Exercice 1 *Les jours en allemand*

Pour commencer, voici le code d'une fonction dont la chaîne de description a été malencontreusement égarée.

```
1  def capitalise(m):
2      assert type(m) == str
3      if m == "":
4          return ""
5      else:
6          return m[0].upper() + m[1:]
```

Question 1 En étudiant soigneusement le code de cette fonction, donnez la chaîne de description de cette fonction. (la méthode `upper` renvoie une chaîne identique à celle à laquelle elle s'applique, sauf pour les lettres qui sont mises en majuscules.)

Ensuite, on donne la ligne de Python suivante :

```
1  cclndjfspdm="di-".join( ["lun","mar","mercre","jeu","vendre","same","dimanche"])
```

Question 2 Précisez le type de la variable `cclndjfspdm`, et décrivez son contenu par une phrase.

Dans la suite, on suppose définie la variable suivante :

```
1  jf="lundi-mardi-mercredi-jeudi-vendredi-samedi-dimanche"
```

Question 3 Quel est l'effet de l'instruction `lf=jf.split("-")` ?

La liste suivante a été définie :

```
1  ld=["montag","dienstag","mittwoch","donnerstag","freitag","samstag","sonntag"]
```

Il s'agit de la liste des noms des jours en allemand. Néanmoins en allemand les noms communs doivent commencer par une lettre capitale.

Question 4 Donnez **une expression** permettant de construire à partir de `ld` la liste des noms des jours en allemand correctement écrits.

Question 5 En utilisant certaines variables précédemment définies, construisez le dictionnaire suivant (évidemment sans réécrire les noms des jours) :

```
1  dico={ "Montag" : "lundi",
2         "Dienstag" : "mardi",
3         "Mittwoch": "mecredi",
4         "Donnerstag": "jeudi",
```

```
5           "Freitag": "vendredi",
6           "Samstag": "samedi",
7           "Sonntag":"dimanche"}
```

Question 6 Donnez le code python d'un prédicat nommé `ist_Tag` qui teste si la chaîne passée en paramètre est un nom de jour en allemand.

Question 7 Donnez le code python d'une fonction nommée `traduction_jour` qui prend en paramètre un nom de jour en allemand et qui renvoie le nom du jour en français. Une exception doit se déclencher si la chaine passée en paramètre n'est pas un nom de jour en allemand.

Question 8 Réalisez une fonction `inverse` paramétrée par un dictionnaire `d` dont on suppose que toutes les valeurs sont des chaines différentes et dont le résultat est le dictionnaire nd dont les cles sont les valeurs de d et dont les valeurs associées sont les clés de `d`, par exemple `inverse(dico)` donne :

```
1  {'mardi'    : 'Dienstag',
2   'samedi'   : 'Samstag',
3   'vendredi' : 'Freitag',
4   'jeudi'    : 'Donnerstag',
5   'lundi'    : 'Montag',
6   'dimanche' : 'Sonntag',
7   'mecredi'  : 'Mittwoch'}
```

Exercice 2 *Sur les fichiers*

Un fichier texte nommé `agenda.csv` contient des numéros de téléphone. Chaque ligne de ce fichier donne : le nom, le prénom, le sexe, et le numéro de téphone de la personne, chacun de ces champs étant séparés par des points-virgules (;). Voici un petit aperçu de quelques lignes d'un tel fichier :

```
Calbuth;Raymond;M;0612345678
Cru;Carmen;F;+33712345678
Talon;Achille;M;+32687654321
```

Question 1 Donnez les quelques lignes de code qui permettent d'attribuer à une variable nommée `nbre_lignes` le nombre de lignes que contient ce fichier,

1. en supposant que l'intégralité du fichier peut être lue en mémoire ;
2. en supposant que la mémoire n'est pas suffisante pour une lecture intégrale du fichier.

Question 2 Le fichier contient des filles et des garçons. Le troisième champ de chaque ligne donne ce sexe : `M` pour les garçons et `F` pour les filles. Séparez-les : les filles dans un fichier nommé `agenda_F.csv`, les garçons dans `agenda_G.csv`.

Exercice 3

On doit gérer les notes d'un groupe d'étudiants dans différentes UE. Les étudiants sont identifiés par un numéro unique : leur NIP (numéro à 7 chiffres commençant par 1). Les UE aussi ont un numéro unique à 6 chiffres commençant par 4.

Voici à titre d'exemple un groupe de trois étudiants qui suivent deux UE (enfin pas tous) et leurs notes.

```
etudiants = {
    1123211 : {'nom' : 'Calbuth', 'prénom' : 'Raymond'},
    1123212 : {'nom' : 'Cru',     'prénom' : 'Carmen'},
    1123213 : {'nom' : 'Talon',   'prénom': 'Achille'}
}
ues = {
    405678 : {'UE': 'Info', 'Intitulé' : 'Informatique',                  'coeff' : 4},
    405679 : {'UE': 'AP1',  'Intitulé' : 'Algorithmes et Programmation 1', 'coeff' : 5}
}
```

```
notes = {
    (1123211, 405678) : 12.5,
    (1123212, 405679) : 13.0,
    (1123212, 405678) : 14.5,
    (1123213, 405679) : 15.5,
    (1123213, 405678) : 18.5
}
```

Question 1 Dans cet exemple, quel est l'étudiant à qui il manque une note, et laquelle ?

Question 2 On donne le code de la fonction suivante :

```
def listenotes(nipetudiant):
    l = []
    for e, u in note:
        if e == nipetudiant:
            l.append(note[(e, u)])
    return l
```

Représentez dans un tableau le contenu de `e`, `u` et `l` pour un appel avec `nipetudiant`=1123212.

Question 3 Réalisez une fonction qui donne la liste des couples `(codeUE,noteUE)` d'un étudiant dont on passe le NIP en paramètre.

Question 4 Réalisez une fonction qui renvoie la moyenne générale coefficientée d'un étudiant dont on passe le NIP en paramètre. S'il manque une note, le résultat renvoyé est `None`

Question 5 Réalisez une fonction qui affiche le relevé de notes pour un étudiant passé en paramètre selon le format ci-dessous :

```
Nom : Talon  Prénom: Achille  NIP: 1123213

405678 Info 5 18.5
405679 AP1 4 15.5

Moyenne générale: 17.17
```

Exercice 4 *Pokemon*

Sur Wikipedia, on peut trouver la liste des 720 noms de Pokemon. Le but de cet exercice est de construire une liste la plus longue possible de noms de Pokemon **tous différents**, telle que, hormis le dernier, chaque nom dans cette liste a sa **dernière lettre identique à la première du mot suivant**. Voici une telle liste :
bulbizarre, empiflor, roucoul, leveinard, dardagnan, nidoqueen, noeubnoeuf, florizarre.
Dans cet exercice, vous pourrez considérer que les noms sont tous écrits en lettres minuscules non accentuées.

Question 1 Écrivez une fonction nommée `suite_valide` qui retourne `True` si la liste de noms passée en paramètre respecte la contrainte décrite ci-dessus, et `False` sinon.

Question 2 Réalisez une fonction qui à partir de la liste des noms de Pokemon renvoie une liste de noms respectant cette contrainte, les choix étant fait au hasard, la construction de la liste s'arrêtant lorsqu'elle ne peut plus être prolongée.
