Examen de Langages de script

Numéro: _____

Consignes Vous avez 2 heures pour compléter l'examen. Si vous manquez de place pour inscrire vos réponses sur ce sujet, indiquez-le et inscrivez la suite sur une copie. Merci de respecter les consignes suivantes :

- Tous les appareils électroniques doivent être éteints et rangés dans votre sac placé à l'avant de la salle. Tous les documents papiers sont autorisés.
- Tous les programmes sont à écrire en PYTHON.
- Il n'est pas nécessaire de préciser les imports.

Exercice 1 :

Écrire une fonction `listes` qui étant donné un dictionnaire associant à des entiers des listes d'entiers renvoie une liste triée sans duplication de tous les entiers contenus dans les listes.
Par exemple `listes({1:[1,2,2],3:[2,3,4,3],2:[4,1,5]})` renvoie `[1,2,3,4,5]`.
Préférez un code court.

```
def listes(dico) :
    return sorted(list({y  for x in dico.values() for y in x}))
```

Exercice 2 :

On considère la définition de a suivante:

```
a = ([1,[2,3],[4]],[[5,6],[7],10,11,12])
```

Indiquer ce qui est affiché par les commandes suivantes: (indiquer "Erreur", si une erreur se produit).

`print(2*a[0])`	
`print(a[2])`	
`print(a[0][0:2])`	
`print(a[2][1][0])`	
`print(a[0][1]-a[1])`	
`print(a-a)`	
`print(a[1][0:1:2])`	
`print(a[0][1:-1])`	

Exercice 3 :

Dans cet exercice, vous devez compléter le code de deux classes, `Biblio` et `Reference`. Une `Reference` est une référence bibliographique. Les attributs `author`, `title`, `pages` contiennent des chaînes de caractères correspondant à l'auteur, au titre et aux pages de la référence. La longueur `len(r)` d'une `Reference r` est le nombre de pages. Par exemple, si les pages sont `'12 22'`, cette référence a 11 pages.

On veut exécuter le code suivant.

```
r1 Reference('A1','Titre2','25 40')
r2=Reference('A2','Titre1','41-62')
print(r1<r2)
bib=Biblio([r1,r2])
l = [len(ref) for ref in bib if len(ref)>20]
print(l)
print(str(len(l))+' references parmi '+str(len(bib))+' font plus de 20 pages')
print(l)
```

Université Paris Diderot L3 Informatique Année 2015–2016

1. Complétez au besoin le code de la classe `Reference` et écrivez la classe `Biblio`. Votre code doit contenir les méthodes nécessaires pour que le segment de code ci-dessus s'exécute correctement.

```
class Reference:
  def __init__(self, au,ti,pp):
    self.author=au
    self.title=ti
    self.pages= pp
  def __str__(self):
    return self.author+', '+self.title+',  pp.'+self.pages
  def __lt__(self, other):
    return self.title < other.title if self.author == other.author \
       else self.author < other.author

SOLUTION

  def __len__(self):
    (deg,end)=self.pages.split('-')
    return int(end)-int(deg)-1
class Biblio:
  def __init__(self,lr):
    self.lr=lr
    self.ctr=0
  def __iter__(self):
    self.ctr=0
    return self
  def __next__(self):
    if self.ctr>=len(self.lr):
       self.ctr=0
       raise StopIteration
    self.ctr+=1
    return self.lr[self.ctr-1]
  def __len__(self) :
    return len(self.lr)
```

2. Donner le résultat de l'affichage.

```
SOLUTION: True  [22]  1 references parmi 2 font plus de 15 pages  [22]
```

Exercice 4 :

Donnez la valeur des expressions suivantes (données dans les colonnes) après l'exécution des instructions suivantes (dans les lignes). S'il y a une erreur, lorsque l'expression est évaluée, indiquez Erreur dans la case correspondante. On suppose que le module `re` est importé. On rappelle que `group(0)` correspond au "match" entier.

Instruction	m.group(0)	m.group(1)	m.group(2)	m.group(3)
s1 "BCDEFGH ABCEF" r1=r"([ABC]{3})" m=re.search(r1, s1)				
s2="ABABEFGH ABABCEF" r2=r"([ABC]{2})(\1)" m=re.match(r2, s2)				
s3 "ababbbababcbeabbabe" r3=r"([ab]*).*\1" m=re.match(r3, s3)				
s4="ababbbababcbeabbabe" r4=r"(b[ab]*)\1(.*)\2" m re.search(r4, s4)				
s5="cceefabcedde" r5=r"([ce]*)([^e]*)([de]+)$" m=re.match(r5, s5)				
s6="cesttresfacile" r6=r"tres?" m re.search(r6, s6)				

Exercice 5 :

Une URL est une adresse web qui est composée des parties suivantes, dans l'ordre:
- un protocole (ftp, http, https) suivi de "://" (Ex: http://);
- un nom de domaine (Ex: www.toto.fr);
- éventuellement un numéro de port précédé de ":" (Ex: :8080);
- éventuellement un chemin, précédé de "/" (Ex: /code/script.php);
- éventuellement une liste de paramètres, précédée de "?". Chaque paramètre est donné sous la forme variable=valeur et les paramètres sont séparés par des "&". (Ex: ?q=python&lang=fr). Le nombre de paramètres est illimité.

1. Donner un appel à une méthode du module re qui permet de récupérer les différentes parties d'une URL avec le code suivant:

```
    # votre code ici :

SOLUTION:
    expr r'(http|ftp|https)://([\w.]+)(:(\d{4}))?(/([\d\w.]+))?(\?([\w\d \&]+))?'
    m=re.match(expr, url)


    # completez les indices de groupes :
SOLUTION:
    protocole = m.group(1)
    domaine = m.group(2)
    port = m.group(4)
    chemin = m.group(6)
    param = m.group(8)
```

L'URL est donnée dans une chaîne varurl. Si une des parties est absente de la chaine alors la variable correspondante sera None ou la chaîne vide. Par exemple, si varurl = 'http://www.toto.fr?p1=moto&p2=bateau', alors les valeurs de variables seront comme suit: protocole = 'http', domaine = 'www.toto.fr', port = None, chemin = None, param = 'p1=moto&p2=bateau'.

Votre expression régulière doit filtrer un maximum d'adresses mal formées.

2. Écrire une ligne de code qui crée un dictionnaire dictparam qui associe à chaque variable sa valeur, à partir de la chaîne param de la question ci-dessus. Par exemple, si param = 'p1=moto&p2=bateau', alors le dictionnaire dictparam contient les associations 'p1':'moto' et 'p2':'bateau'.

```
SOLUTION




dictparam = v[0]:v[1] for v in ([ x.split("=") for x in param.split("&") if x] if param else [])
```

Exercice 6 :

Le but de cet exercice est d'écrire un script PYTHON synchro de synchronisation simpliste entre deux arborescences d'un système de fichiers dont les répertoires racines sont passés en argument du script. Étant donné une arborescence de racine A et une copie récursive de A de racine B, si des fichiers ordinaires de A sont modifiés par la suite, alors synchro A B copie ces fichiers dans B. Par exemple, si le fichier ordinaire A/X/f a été modifié, alors synchro A B copie ce fichier dans B/X/ sous le nom f.

Dans cet exercice, les répertoires A et B sont tels que:
- l'arborescence de racine A ne contient que des répertoires et fichiers ordinaires.
- l'arborescence de racine B est une copie de celle de racine A à un instant t.
- dans l'arborescence de racine A, les seules modifications effectuées après l'instant t sont sur le contenu des fichiers ordinaires.

Lorsque vous ajouterez du code à un code déjà écrit, précisez bien la position du code ajouté dans le code déjà écrit (imbriqué dans un bloc d'instructions ou pas, lequel ...).

1. Écrire un script qui étant donné une arborescence de racine A passé en argument du script, crée un dictionnaire où:
   - les clés sont les références (relatives au répertoire courant) des répertoire de l'arborescence de racine A.
   - la valeur correspondant à une clé C est un 2-uplet dont le premier élément est l'ensemble des fichiers ordinaires de C et le deuxième élément est l'ensemble des sous-répertoires de C.

   Par exemple :

```
$ ls -R
X Y
A/X:
f g
A/Y:
Z h
A/Y/Z:
i
```

   alors le script produit le dictionnaire:

```
{'A':({},{'A/X', 'A/Y'}), 'A/X':({'A/X/f','A/X/g'},{}), 'A/Y':({'A/Y/h'},{'A/Y/Z'}),
 'A/Y/Z':({'A/Y/Z/i'},{})}
```

```
#!/usr/bin/python3

d={}
for ch, reps, fics in os.walk(sys.argv[1], topdown=False):
    d[ch] = ({os.path.join(ch,fic) for fic in fics},\
             {os.path.join(ch, rep) for rep in reps})
```

2. Ajouter au script une fonction `synch` qui prend en argument deux répertoires A et B et un dictionnaire d, construit comme ci-dessus à partir de A, et évalue si les fichiers ordinaires de A sont différents de ceux de B.

   Pour cela vous devez utiliser la commande UNIX `diff`. Si f et g sont deux fichiers ordinaires, `diff f g` n'affiche rien si f et g ont un contenu identique et affiche sur la sortie standard les différences s'il y en a.

   Si par exemple, le contenu de `A/X/f` est différent de celui de `B/X/f`, la fonction `synch` doit afficher `A/X/f et B/X/f sont differents` et rien sinon. Aucun autre type de message ne doit s'afficher sur la sortie standard pendant l'exécution de la fonction `synch`.

```
def synchro(A, B, d):
    for fic in d[A][0]:
        fic2 = os.path.join(B, os.path.split(fic)[1])
        cmd = ["diff", fic, fic2]
        p = subprocess.Popen(cmd, stdin=subprocess.PIPE, stdout=subprocess.PIPE,\
            stderr=subprocess.PIPE, universal_newlines=True)
        out, err = p.communicate()
        if not out == None:
            print(fic + " et " + fic2 + " sont differents")
```

3. Ajouter à `synch` le code qui effectue les copies nécessaires de A vers B. Les copies doivent s'effectuer avec la commande UNIX `cp`.

   Dans le code précédent, remplacer l'instruction contenant le `print` par les instructions:

```
cmd = ["cp", fic, fic2]
subprocess.Popen(cmd)
```

4. Ajouter à `synch` le code pour que la synchronisation ne s'effectue pas seulement sur les répertoires A et B, mais sur les arborescences de racines respectives A et B.

   Ajouter après la boucle `for`:

```
for repA in d[A][1]:
        repB = os.path.join(B, os.path.split(repA)[1])
        synchro(repA, repB, d)
```

5. Compléter votre script pour qu'il effectue la synchronisation des deux arborescences dont les racines sont passées en argument du script.

```
synchro(sys.argv[1], sys.argv[2], d)
```