# Examen de Langages de script n° 1 : 2011/2012

---

- Durée : 2h
- Vous devez éteindre et ranger vos téléphones.
- Les programmes sont à faire en PYTHON 3.
- **Chaque exercice doit être rédigée sur une copie distincte.**
- L'annexe du sujet contient
  - des rappels de PYTHON ;
  - la documentation du module `re`.

---

**Exercice 1 :**

1. Écrire une fonction `filterdiv(l, i)` qui prend en argument une liste d'entiers `l` et un entier `i` strictement positif, et qui renvoie la sous-liste de `l` contenant les éléments de `l` qui sont divisibles par `i`.
2. Écrire une fonction `cprod(m, n)` qui prend deux ensembles `n`, `m` en argument, et qui renvoie l'ensemble de tous les couples `(x,y)` formés d'un élément `x` de `m` et un élément `y` de `n`.
3. Écrire une fonction `listn(m, i)` qui prend un ensemble `m` en argument, et renvoie l'ensemble de toutes les listes d'élements de `m` de longueur `i`.
4. Qu'imprime le programme suivant ?

```
d = {i:i*i for i in range(2,-3,-1)}
e = {}
for i in range(2,-3,-1):
    e[d[i]]=i
for i in range(-5,5):
    if i in d: print(d[i])
    if i in e: print(e[i])
```

5. Qu'imprime le programme suivant ?

```
def f(a): print (a); print('f('+chr(34)+a+chr(34)+')')
f("def f(a): print (a); print('f('+chr(34)+a+chr(34)+')')")
```

6. Qu'imprime le programme suivant ?

```
def shuffle(a, b, c = []):
    if a == []:
        return [c + b]
    if b == []:
        return [c + a]
    return shuffle(a[1:], b, c + [a[0]]) + shuffle(a, b[1:], c + [b[0]])
print(shuffle([1,2],[3,4]))
```

   En général, qu'est-ce que fait la fonction `shuffle` appelée avec deux listes ?

7. Écrire une fonction qui étant donné un entier positif retourne une chaîne de caractères contenant la représentation de l'entier avec une virgule entre chaque groupe de 3 chiffres. Par exemple pour 1234, la fonction renvoie la chaîne de caractères `"1,234"`, pour 3 elle renvoie `"3"` et pour 1234567 elle renvoie `"1,234,567"`.

**Exercice 2 :**

Le but de cet exercice est de gérer des SMS. Pour traiter les questions qui suivent il est recommandé de lire l'exercice en entier et d'introduire toutes les fonctions auxiliaires que vous jugerez nécessaire en expliquant leur rôle.

Un fichier de SMS est un texte de la forme suivante :

```
0654342310 31/05/2012 10h23 Salut, t'es ou ?
0789304059 30/05/2012 11h10 Carla t'a laissé un message. Rappelle le : 0899432340
0654394503 01/06/2012 09h03 T'as pas la solution de l'exo 2 ?
0654394503 01/06/2012 09h40 Qu'est-ce que tu fais, y en a marre d'attendre.
0660324524 29/05/2012 06h30 Rdv dans 5 min au 0899 bvd Royal.
0653434325 28/04/2012 03h20 Le nouveau IPHONE à 400$ chez phonehouse.
0143523523 03/06/2012 16h23 ADIDAS: PLUS QUE 200 MODELES DISPO CHEZ DECATHLON.
```

Chaque ligne contient exactement un SMS. Les numéros de téléphone ont tous 10 chiffres, les dates sont toujours formées de 10 caractères et l'heure de 5. Des SMS seront représentés comme un dictionnaire. Les clefs sont les numéros de téléphone d'origine et les valeurs des listes de messages. Chaque message est une liste de 3 éléments [date,heure,contenu]. Choisissez une représentation adéquate pour la date et l'heure.

On considère qu'un SMS est publicitaire s'il contient le caractère $ **ou** si plus que la moitié des caractères sont des majuscules. On considère qu'un message est un SPAM s'il contient un numéro de téléphone commençant avec 0899 (un numéro de téléphone a 10 chiffres).

Pour répondre à cet exercice, vous devez rédiger d'une part un module python nommé `sms.py` et d'autre part donner les explications nécessaires à la compréhension de votre code.

1. Écrire une fonction qui étant donné comme argument le nom d'un fichier de SMS renvoie un dictionnaire de SMS.
2. Écrire une fonction qui étant donné un dictionnaire de SMS, renvoie un dictionnaire sans les SMS SPAM.
3. Écrire une fonction qui élimine d'un dictionnaire de SMS tous les SMS publicitaires.
4. Écrire une fonction qui extrait d'un dictionnaire de SMS une "liste noire" de numéros de téléphone qui ont envoyé soit des SMS publicitaires soit des SMS SPAM.
5. Écrire une fonction qui permet de chercher un mot dans un dictionnaire SMS. Elle doit renvoyer tous les numéros qui ont envoyé un message contenant le mot. Attention : Si on cherche "la" le deuxième numéro n'est pas renvoyé.
6. Écrire une fonction qui étant donné un dictionnaire de SMS et une date renvoie un dictionnaire de SMS sans les SMS qui sont plus vieux que 2 semaines.
7. Le module `sms.py` se terminera par la partie principale du code qui ne sera exécutée que lorsque le module est lancé en ligne de commande à partir d'un terminal unix avec comme paramètre le nom d'un fichier SMS. Cette partie principale imprimera sur l'écran tous les SMS non publicitaires.
8. Décrire les opérations à effectuer afin que la commande `./sms` lance le script correspondant.

# Annexe

## a) Rappel de quelques éléments de Python

– `range(i,j,l)` permet de parcourir les entiers de `i` à `j` exclu avec un pas de `l`.

```
>>> for i in range(3,-4,-1):
...     print(i)
...
3
2
1
0
-1
-2
-3
```

– La fonction `chr` prend en argument un entier et renvoie le caractère correspondant dans le code `ASCII`.

```
>>> print(chr(34))
...
...
"
```

– Le point-virgule « ; » permet de séparer plusieurs instructions s'enchaînant sur une même ligne.

```
>>> y = 5; x = 2*y; print(x)
...
...
10
```

## b) Module `re` - descriptif basé sur le livre de Harold Erbin

**Syntaxe** Les regex répondent à une syntaxe très codifiée et possèdent de nombreux symboles ayant un sens particulier. Pour débuter, tout caractère alphanumérique n'a pas de signification spéciale : A correspond simplement à la lettre A, 1 au chiffre 1, etc. Quant aux principaux symboles spéciaux, il sont :

`.` : désigne n'importe quel caractère ;

`^` : indique que le début de la chaine doit correspondre ;

`$` : indique que la fin de la chaine doit correspondre ;

`{n}` : indique que le caractère précédent doit être répété n fois.

`{n,m}` : indique que le caractère précédent doit être répété entre n et m fois.

`*` : le caractère précédent peut être répété aucune ou plusieurs fois. Par exemple, à ab* peuvent correspondre : a, ab, ou a suivi d'un nombre quelconque de b.

`+` : le caractère précédent peut être répété une ou plusieurs fois. Par exemple, à ab+ correspond un a suivi d'un nombre quelconque de b.

`?` : le caractère précédent peut être répété zéro ou une fois. Par exemple, à ab ? correspondent ab et a.

L'antislash permet d'échapper tous ces caractères spéciaux. Les crochets [] permettent d'indiquer une plage de caractère, par exemple [e-h] correspondra à e, f, g ou h. Finalement, il reste quelques caractères spéciaux assez utiles :

`\w` : il correspond à tout caractère alphabétique, c'est à dire qu'il est équivalent à [a-zA-Z] ;

`\W` : il correspond à tout ce qui n'est pas un caractère alphabétique ;

`\b` : il correspond à la frontière (début ou fin) d'un mot ;

`\d` : il correspond à tout caractère numérique, c'est à dire qu'il est équivalent à [0-9] ;

`\D` : il correspond à tout ce qui n'est pas un caractère numérique.

### Utilisation

`re.search(pattern, string)` Cherche le motif dans la chaîne passée en argument et retourne un `MatchObject` si des correspondances sont trouvées, sinon retourne `None`.

`re.split(pattern, string)` Découpe la chaine string selon les occurences du motif.

```
>>> re.split(r'\W', 'Truth is beautiful, without doubt.')
['Truth', 'is', 'beautiful', '', 'without', 'doubt', '']
```

`re.findall(pattern, string)` Retourne toutes les sous-chaines de `string` correspondant au motif.

`re.sub(pattern, repl, string)` Retourne la chaîne `string` où le motif a été remplacé par `repl`.