INFO-H-100 - Informatique

Séance d'exercices 14
Introduction à Python
Map, Filter, Reduce, Lambda

Université Libre de Bruxelles
Faculté des Sciences Appliquées

10 février 2014

Map

`map` est une fonction qui reçoit comme argument une liste `li` et une fonction `fn`, et qui renvoie comme résultat la liste composée des résultats de l'application de la fonction `fn` sur chaque élément de la liste `li`.

Plus concrètement :

```
def my_map(fn, li):
    res = []
    for elem in li:
        res.append(fn(elem))
    return res
```

Attention, la fonction `map` renvoie un objet de type `map`. Utilisez la fonction `list` pour convertir cet objet sous la forme d'une liste.

Exemple :

```
>>> list(map(float, [1, 2, 3]))
[1.0, 2.0, 3.0]
```

Filter

Filter est une fonction qui reçoit en argument une fonction `fn` et une liste `li`, et qui renvoie une liste composée des éléments `elem` de `li` pour lesquels `fn(e)` est vrai.

```
def my_filter(fn, li):
    res = []
    for elem in li:
        if fn(elem):
            res.append(elem)
    return res
```

Attention, la fonction `filter` renvoie un objet de type `filter`. Utilisez la fonction `list` pour convertir cet objet sous la forme d'une liste.

Exemple :

```
>>> list(filter(str.isdigit, ["bonjour", "1", "45", "a"]))
["1", "45"]
```

Reduce

Reduce est une fonction qui reçoit en argument une fonction `fn` et une liste `li`, et qui va calculer le résultat de l'appel `fn(fn(fn(...fn(fn(li[0], li[1]), li[2])...)))`

```
def my_reduce(fn, li):
    res = li[0]
    for elem in li[1:]:
        res = fn(res, elem)
    return res
```

Exemple :

```
>>> def my_add(a, b): return a + b
>>> functools.reduce(my_add, [1, 2, 3])
6
```

Lambda

L'opérateur `lambda` permet de définir des fonctions à la volée, sans leur donner de nom. Par exemple :

```
>>> map(lambda x: x+1, [1, 2, 3])
[2, 3, 4]
```

Les deux codes suivants sont équivalents :

```
mult = lambda x, y: x * y
```

```
def mult(x, y): return x * y
```

Exemple de lambda : Le tri

```
def insertion_sort(ls,pred_fn):
    for i in range(1, len(ls)):
        val = ls[i]
        j=i
        while j > 0 and pred_fn(val, ls[j - 1]):
            ls[j] = ls[j - 1]
            j -= 1
        ls[j] = val
```

```
>>> ls = [2, 5, 9, 9, 6, 6, 4, 5, 2, 3, 1, 5, 8]
>>> insertion_sort(ls, lambda x,y: x < y)
>>> ls
[1, 2, 2, 3, 4, 5, 5, 5, 6, 6, 8, 9, 9]
```

```
>>> ls = [{"name" : "Jean Dupont", "city" : (4500,"Tihange")},
          {"name" : "Pierre Leloup", "city" : (1000,"Bruxelles")},
          {"name" : "Jacques du Four", "city" : (1050,"Ixelles")},
          {"name" : "Kim Hainaut", "city" : (4000,"Liege")}]
>>> insertion_sort(ls, lambda x,y: x["city"][0] < y["city"][0])
```