

MOOC Python

Corrigés de la semaine 3

comptage - Semaine 3 Séquence 1

```
1 def comptage(in_filename, out_filename):
2     """
3     retranscrit le fichier in_filename dans le fichier out_filename
4     en ajoutant des annotations sur les nombres de lignes, de mots
5     et de caractères
6     """
7     # on ouvre le fichier d'entrée en lecture
8     # on aurait pu mettre open(in_filename, 'r')
9     with open(in_filename, encoding='utf-8') as input:
10         # on ouvre la sortie en écriture
11         with open(out_filename, "w", encoding='utf-8') as output:
12             # initialisations
13             total_words = 0
14             total_chars = 0
15             # pour toutes les lignes du fichier d'entrée
16             # le numéro de ligne commence à 1
17             for lineno, line in enumerate(input, 1):
18                 # autant de mots que d'éléments dans split()
19                 nb_words = len(line.split())
20                 total_words += nb_words
21                 # autant de caractères que d'éléments dans la ligne
22                 nb_chars = len(line)
23                 total_chars += nb_chars
24                 # on écrit la ligne de sortie; pas besoin
25                 # de newline (\n) car line en a déjà un
26                 output.write("{}: {}: {}:{}"
27                               .format(lineno, nb_words, nb_chars, line))
28             # on écrit la ligne de synthèse
29             # lineno est une variable de boucle, elle "fuite"
30             # on peut donc utiliser sa dernière valeur
31             # mais remarquez que ce code ne fonctionnerait
32             # pas sur un fichier vide, ou on aurait lineno non définie
33             output.write("{}: {}: {}{}\n"
34                           .format(lineno, total_words, total_chars))
```

```
1 def index(bateaux):
2     """
3     Calcule sous la forme d'un dictionnaire indexé par les ids
4     un index de tous les bateaux présents dans la liste en argument
5     Comme les données étendues et abrégées ont toutes leur id
6     en première position on peut en fait utiliser ce code
7     avec les deux types de données
8     """
9     # c'est une simple compréhension de dictionnaire
10    return {bateau[0] : bateau for bateau in bateaux}
```

```
1 def index_bis(bateaux):
2     """
3     La même chose mais de manière itérative
4     """
5     # si on veut décortiquer
6     resultat = {}
7     for bateau in bateaux:
8         resultat [bateau[0]] = bateau
9     return resultat
```

```
1 def merge(extended, abbreviated):
2     """
3     Consolide des données étendues et des données abrégées
4     comme décrit dans l'énoncé
5     Le coût de cette fonction est linéaire dans la taille
6     des données (longueur commune des deux listes)
7     """
8     # on initialise le résultat avec un dictionnaire vide
9     result = {}
10    # pour les données étendues
11    # on affecte les 6 premiers champs
12    # et on ignore les champs de rang 6 et au delà
13    for id, latitude, longitude, timestamp, name, country, *ignore in extended:
14        # on crée une entrée dans le résultat,
15        # avec la mesure correspondant aux données étendues
16        result[id] = [name, country, (latitude, longitude, timestamp)]
17    # maintenant on peut compléter le résultat avec les données abrégées
18    for id, latitude, longitude, timestamp in abbreviated:
19        # et avec les hypothèses on sait que le bateau a déjà été
20        # inscrit dans le résultat, donc result[id] doit déjà exister
21        # et on peut se contenter d'ajouter la mesure abrégée
22        # dans l'entrée correspondant dans result
23        result[id].append((latitude, longitude, timestamp))
24    # et retourner le résultat
25    return result
```

```
1 def merge_bis(extended, abbreviated):
2     """
3     Une deuxième version, linéaire également
4     """
5     # on initialise le résultat avec un dictionnaire vide
6     result = {}
7     # on remplit d'abord à partir des données étendues
8     for ship in extended:
9         id = ship[0]
10        # on crée la liste avec le nom et le pays
11        result[id] = ship[4:6]
12        # on ajoute un tuple correspondant à la position
13        result[id].append(tuple(ship[1:4]))
14    # pareil que pour la première solution,
15    # on sait d'après les hypothèses
16    # que les id trouvées dans abbreviated
17    # sont déjà présentes dans le resultat
18    for ship in abbreviated:
19        id = ship[0]
20        # on ajoute un tuple correspondant à la position
21        result[id].append(tuple(ship[1:4]))
22    return result
```

```

1 def merge_ter(extended, abbreviated):
2     """
3     Une troisième solution
4     à cause du tri que l'on fait au départ, cette
5     solution n'est plus linéaire mais en  $O(n \cdot \log(n))$ 
6     """
7     # ici on va tirer profit du fait que les id sont
8     # en première position dans les deux tableaux
9     # si bien que si on les trie,
10    # on va mettre les deux tableaux 'en phase'
11    #
12    # c'est une technique qui marche dans ce cas précis
13    # parce qu'on sait que les deux tableaux contiennent des données
14    # pour exactement le même ensemble de bateaux
15    #
16    # on a deux choix, selon qu'on peut se permettre ou non de
17    # modifier les données en entrée. Supposons que oui:
18    extended.sort()
19    abbreviated.sort()
20    # si ça n'avait pas été le cas on aurait fait plutôt
21    # extended = extended.sorted() et idem pour l'autre
22    #
23    # il ne reste plus qu'à assembler le résultat
24    # en découpant des tranches
25    # et en les transformant en tuples pour les positions
26    # puisque c'est ce qui est demandé
27    return {
28        e[0] : e[4:6] + [ tuple(e[1:4]), tuple(a[1:4]) ]
29        for (e,a) in zip (extended, abbreviated)
30    }

```

```

1 from collections import defaultdict
2
3 def parse_graph(filename):
4     g = defaultdict(list)
5     with open(filename) as f:
6         for line in f:
7             begin, value, end = line.split()
8             g[begin].append( (end, int(value)))
9     return g

```

```

1 def diff(extended, abbreviated):
2     """Calcule comme demandé dans l'exercice, et sous formes d'ensembles
3     (*) les noms des bateaux seulement dans extended
4     (*) les noms des bateaux présents dans les deux listes
5     (*) les ids des bateaux seulement dans abbreviated
6     """
7     ### on n'utilise que des ensembles dans tous l'exercice
8     # les ids de tous les bateaux dans extended
9     # une compréhension d'ensemble
10    extended_ids = {ship[0] for ship in extended}
11    # les ids de tous les bateaux dans abbreviated
12    # idem
13    abbreviated_ids = {ship[0] for ship in abbreviated}
14    # les ids des bateaux seulement dans abbreviated
15    # une difference d'ensembles
16    abbreviated_only_ids = abbreviated_ids - extended_ids
17    # les ids des bateaux dans les deux listes
18    # une intersection d'ensembles
19    both_ids = abbreviated_ids & extended_ids
20    # les ids des bateaux seulement dans extended
21    # ditto
22    extended_only_ids = extended_ids - abbreviated_ids
23    # pour les deux catégories où c'est possible
24    # on recalcule les noms des bateaux
25    # par une compréhension d'ensemble
26    both_names = \
27        {ship[4] for ship in extended if ship[0] in both_ids}
28    extended_only_names = \
29        {ship[4] for ship in extended if ship[0] in extended_only_ids}
30    # enfin on retourne les 3 ensembles sous forme d'un tuple
31    return extended_only_names, both_names, abbreviated_only_ids

```

```

1 def diff_bis(extended, abbreviated):
2     """
3     Idem avec seulement des compréhensions
4     """
5     extended_ids = {ship[0] for ship in extended}
6     abbreviated_ids = {ship[0] for ship in abbreviated}
7     abbreviated_only = {ship[0] for ship in abbreviated
8                          if ship[0] not in extended_ids}
9     extended_only = {ship[4] for ship in extended
10                     if ship[0] not in abbreviated_ids}
11    both = {ship[4] for ship in extended
12           if ship[0] in abbreviated_ids}
13    return extended_only, both, abbreviated_only

```