FEUILLE n°6 *Application du tri*

## Exercice 1. Question de cours

### 1.1. Compléter le code de la fonction `mystere` (documentation et test)

```
def mystere(liste):
    """
    parametre : une liste de str
    resultat: ???
    """
    return max(liste, key = len)

assert mystere(['You', 'know', 'nothing', 'John', 'Snow']) == ???
```

### 1.2. Compléter le script suivant

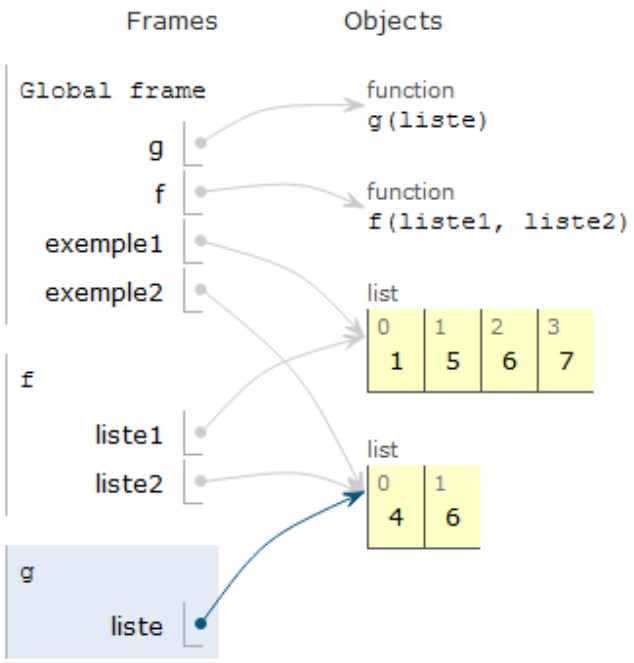
```
def somme(???):
   ???

def tri_selon_la_somme(liste):
    """
    parametre : une liste de couples de nombres
    resultat: la liste triée dans l'ordre croissant de la somme des deux
    composants des couples
    """
    return sorted(liste, key = somme)

assert tri_selon_la_somme([(5, 7), (12, 3), (9, 1), (5, 6)]) == [(9, 1), (5, 6),
    (5, 7), (12, 3)]
```

### 1.3. Compléter le code de la fonction `tri_selon_la_deuxieme_composante`

```
def tri_selon_la_deuxieme_composante(liste):
    """
    parametre : une liste de couples de nombres
    resultat: la liste triée dans l'ordre croissant selon la deuxième composante
    des couples
    """
    ???

assert tri_selon_la_deuxieme_composante([(5, 7), (12, 3), (9, 1), (5, 6)]) == [(9, 1) ,
    (12, 3), (5, 6), (5, 7)]
```

## Exercice 2. Représentation de la mémoire

**2.1.** Proposer un script qui provoque la représentation de la mémoire ci-contre. Préciser à quel endroit dans le script on se trouve.




## Exercice 3. Lecture de code

On représente un intervalle `intervalle = [debut, fin[` par le tuple `(debut, fin)`.

**3.1.** Compléter le code (documentation et tests) de la fonction suivante :

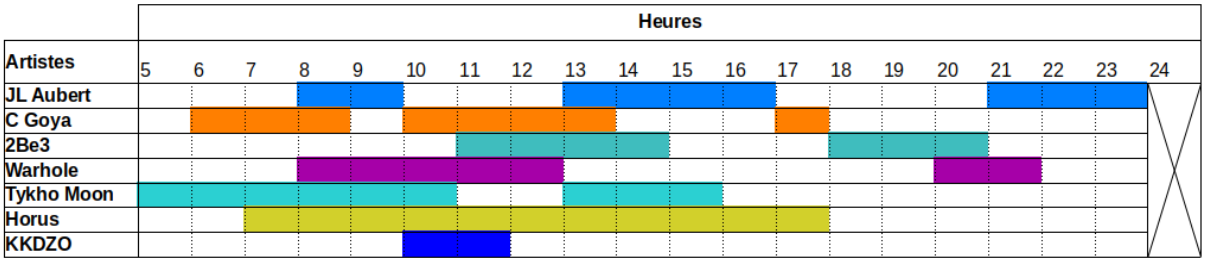
```
def mystere(intervalle1, intervalle2):
    """
    paramètres : deux intervalles [debut, fin[ représentés par des tuples
    résultat : ???
    """
    (debut1, fin1) = intervalle1
    (debut2, fin2) = intervalle2
    return debut1 < fin2 and debut2 < fin1

assert mystere((1, 4), (6, 8)) == ???
assert mystere((7, 9), (2, 4)) == ???
assert mystere((1, 11), (6, 8)) == ???
assert mystere((8, 14), (5, 10)) == ???
```

### Exercice 4. Planning pour un festival : algorithmique sans code

Au cours d'un festival, plusieurs spectacles ont lieu, souvent sur plusieurs scènes. On veut écrire une application pour aider des festivaliers à assister à un maximum de spectacles.

Par exemple, le Festival *Enki Bilal* se déroule tous les ans le 32 décembre dans la ville de Nikopol. Ce festival commence tôt le matin (dès 5h) et se termine à minuit. Les artistes à l'affiche cette année sont les suivants :

- JL Aubert se produira de 8h à 10h. Après une courte pause, il donnera un deuxième spectacle de 13h à 17h. Puis, il reviendra plus tard dans la soirée de 21h à 24h.
- La grande artiste C Goya donnera également trois représentations. La première de 6h à 9h, la deuxième de 10h à 14h et la dernière de 17h à 18h.
- Les 2Be3 viendront nous honorer de leur présence. Ils donneront deux spectacles : l'un de 11h à 15h, et l'autre de 18h à 21h.
- Le groupe de musique local Warhole se produira de 8h à 13h puis de 20h à 22h.
- Le célèbre Tykho Moon donnera deux concerts : le premier de 5h à 11h et le second de 13h à 16h.
- La troupe de théâtre Horus assurera un spectacle continu entre 7h et 18h.
- Enfin, le groupe KKDZO assurera le show de 10h à 12h.




**4.1.** On considère que les spectacles ont lieu pendant l'intervalle de temps `[début, fin[` et on considère que deux spectacles sont compatibles si leur deux intervalles de temps ne se superposent pas. Exhiber deux spectacles compatibles et deux spectacles incompatibles.

**4.2.** Anakin, grand fan du festival Enki Bilal, veut assister au maximum de spectacles possibles. Il prévoit d'arriver dès le début du festival à 5h. Il a fait la pré-sélection suivante : Tykho Moon à 5h, JL Aubert à 13h et les 2Be3 à 18h. Il se demande s'il n'est pas possible d'aller voir plus de spectacles. Pouvez-vous lui proposer une sélection de spectacles plus fournie que la sienne ?

**4.3.** Quelle est, d'après-vous, la sélection de spectacles optimale, c'est-à-dire celle qui comporte le plus grand nombre de spectacle ? (Pas de preuve demandée ici, juste une solution intuitive).

### 4.4. Un premier algorithme

Pour maximiser le nombre de spectacles vus, Anakin décide d'utiliser l'algorithme suivant :

```
Données :
  * programme : tous les spectacles du festival avec leur heure de
                debut et leur heure de fin
Résultat : les spectacles sélectionnés (tous compatibles)
Algorithme :
  * initialiser selection à la liste vide
  * initialiser heure_actuelle à l'heure de début du festival
  * initialiser selection_est_en_cours à True
  * tant que la selection_est_en_cours
      - on choisit dans le programme le premier spectacle qui commence
        après heure_actuelle. Puis :
         + on ajoute ce spectacle à la sélection
         + heure_actuelle prend la valeur de l'heure de fin du spectacle
      - si aucun spectacle ne peut être choisi, selection_est_en_cours
        prend la valeur False.
  * Renvoyer la selection
```

S'il arrive dès le début du festival, quelle sélection lui proposera cet algorithme ?

### 4.5. Un deuxième algorithme

Pour maximiser le nombre de spectacles vus, Anakin a une autre idée : il décide de favoriser les spectacles de petite durée. Proposer un algorithme qui utilise la durée des spectacles comme critère de choix.

Avec cet algorithme, s'il arrive dès le début du festival, quelle sera sa sélection ?

### 4.6. Un troisième algorithme

Anakin a une autre idée : il choisit tour à tour le spectacle compatible avec sa sélection qui se termine le plus tôt possible possible.

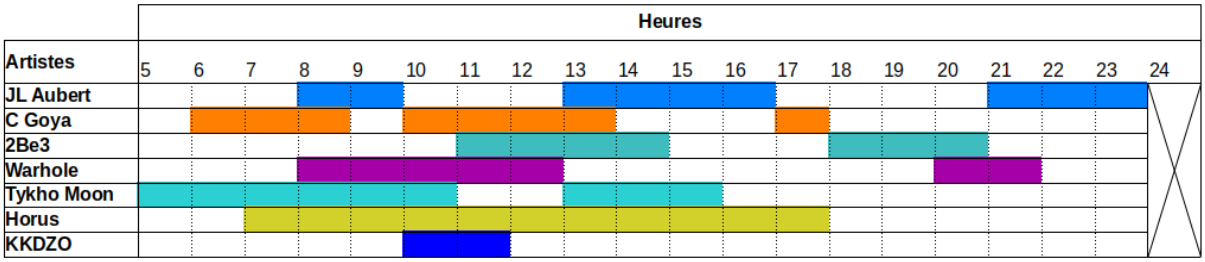
Avec cet algorithme, s'il arrive dès le début du festival, quelle sera sa sélection ?

### 4.7. Réflexion sur un début d'implémentation

a) Quelle structure de données pourrait représenter un `spectacle` ? Donner un exemple de modélisation du premier spectacle de JL Aubert.
b) Quelle structure de données pourrait représenter le programme complet d'un festival ? Donner un exemple de modélisation du programme du festival de Nikopol.
c) Quelle structure de données pourrait représenter une sélection de spectacles ? Donner un exemple avec la sélection de Anakin.

### Exercice 5. Problème : planning pour un festival : implémentation en python

Au cours d'un festival, plusieurs spectacles ont lieu, souvent sur plusieurs scènes. On veut écrire une application pour aider des festivaliers à assister à un maximum de spectacles.




```python
# Exemples de modélisation de spectacles :
s1 = {'nom': 'JL Aubert', 'debut': 8, 'fin': 10}
s2 = {'nom': '2Be3', 'debut': 11, 'fin': 15}
s3 = {'nom': 'Tyko Moon', 'debut': 5, 'fin': 11}

# Exemple de modélisation de programme :
nikopol = [
    {'nom': 'JL Aubert', 'debut': 8, 'fin': 10},
    {'nom': 'JL Aubert', 'debut': 13, 'fin': 17},
    {'nom': 'JL Aubert', 'debut': 21, 'fin': 24},
    {'nom': 'C Goya', 'debut': 6, 'fin': 9},
    {'nom': 'C Goya', 'debut': 10, 'fin': 14},
    {'nom': 'C Goya', 'debut': 17, 'fin': 18},
    {'nom': '2Be3', 'debut': 11, 'fin': 15},
    {'nom': '2Be3', 'debut': 18, 'fin': 21},
    {'nom': 'Warhole', 'debut': 8, 'fin': 13},
    {'nom': 'Warhole', 'debut': 20, 'fin': 22},
    {'nom': 'Tyko Moon', 'debut': 5, 'fin': 11},
    {'nom': 'Tyko Moon', 'debut': 13, 'fin': 16},
    {'nom': 'Horus', 'debut': 7, 'fin': 18},
    {'nom': 'KKDZO', 'debut': 10, 'fin': 12}
    ]
```

**5.1.** Pourquoi ne pas modéliser le programme d'un festival par un ensemble de spectacles ?

**5.2.** Écrire une fonction `compatibles(spectacle1,spectacle2)` qui détermine si les deux spectacles `spectacle1` et `spectacle2` sont compatibles, c'est à dire s'il est possible de les voir tous les deux. [1]

1. Indication : allez voir l'exercice 3

**5.3.** Écrire une fonction `tous_compatibles(selection, spectacle)` qui détermine si le spectacle `spectacle` est compatible avec tous les spectacles de `selection` (`selection` étant une liste de spectacles).

### 5.4. Implémentation du premier algorithme de l'exercice précédent

**a)** Écrire une fonction qui `tri_selon_debut` qui prend en paramètre un programme et qui trie les spectacles du programme selon leur heure de début croissante.

**b)** Écrire une fonction `prochain_spectacle(programme, heure)` qui prend en paramètre un programme dont les spectacles sont **triés** par heure de début croissante et qui donne le premier spectacle qui commence après l'heure indiquée.

**c)** Écrire une fonction `selection1(programme)` qui propose la sélection de spectacles donnée par l'algorithme 1.

### 5.5. Implémentation du deuxième algorithme de l'exercice précédent

**a)** Écrire une fonction qui `tri_selon_duree` qui prend en paramètre un programme et qui trie les spectacles du programme selon leur durée croissante

**b)** Écrire une fonction `prochain_spectacle(programme, selection)` qui prend deux paramètres : `programme` dont les spectacles sont **triés** par heure de début croissante et `selection` qui est une sélection de programmes. Cette fonction doit renvoyer le premier spectacle du programme qui est compatible avec tous les spectacles de la sélection.

**c)** Écrire une fonction `selection2(programme, heure_arrivee)` qui propose la sélection de spectacles donnée par l'algorithme 2.

### 5.6. Implémentation du troisième algorithme de l'exercice précédent

Écrire une fonction `selection3(programme, heure_arrivee)` qui propose la sélection de spectacles donnée par l'algorithme 3.

**5.7.** D'après vous, lequel de ces trois algorithmes est le "meilleur" ?

### Exercice 6. Ranger les malles du Père Noël

Le Père Noël est en plein préparatifs de la grande nuit. Il a modélisé la liste de tous les cadeaux qu'il doit livrer de la façon suivante :

```
# Un cadeau est modélisé par un dictionnaire (son nom et la place qu'il prend)
train = {"nom": "train", "taille": 18}
nours = {"nom": "peluche", "taille": 48}
velo = {"nom": "velo", "taille": 25}
stylo = {"nom": "stylo", "taille": 1}
console = {"nom": "console", "taille": 5}

cadeaux_2020 = [train, nours,  velo, stylo, console]
```

Il faut maintenant ranger tous ces cadeaux dans les malles de son traîneau.

— Une malle est modélisée par une liste de cadeaux. Chaque malle a une contenance maximale de 50, ce qui signifie que les cadeaux qu'elle contient ne peuvent totaliser une taille supérieure à 50.

— Le traîneau sera modélisé par une liste de malles.

**6.1.** Dans un premier temps, les lutins prennent les cadeaux dans l'ordre de la liste et les mettent dans les malles au fur et à mesure. Quand le cadeau qu'ils ont entre les mains ne tient pas dans la malle qui est ouverte, ils ferment cette malle, en ouvrent une nouvelle qu'ils mettent dans le traineau et commencent à la remplir.

Compléter le code de la fonction `range_betement`. Vous pouvez écrire le code de fonctions auxilliaires.

```
def range_betement(liste_des_cadeaux):
    """
    renvoie le traineau contenant tous les cadeaux de la liste rangés dans des malles
    la contenance des malles étant de 50 max
    """
    pass

assert range_betement(cadeaux_2020) == [[train], [nours], [velo, stylo, console]]
```

**6.2.** Au bout d'un moment, les lutins se rendent compte que ce rangement ne semble pas du tout optimisé. Ils pensent qu'il est possible d'utiliser moins de malles mais ils ne savent pas trop comment faire. Pouvez-vous les aider ?