Méthodologie de programmation

Session 3

Alex Singh

Structures de données de base

- Les structures de données sont des moyens d'organiser des collections de données.
- Elles sont souvent accompagnées de fonctions permettant de les manipuler.
- Les structures de données peuvent être assez génériques (pensez aux listes, par exemple) ou conçues pour des applications spécifiques (les tas de Fibonacci, par exemple).

Listes

- Les listes sont l'un des types les plus courants de structures de données « séquentielles ».
- Opérations courantes : insertion, concaténation, suppression, recherche, fold, map.
- Les listes en Python ne doivent pas nécessairement être homogènes : les éléments peuvent être de types différents.
- Nous pouvons itérer sur des listes à l'aide de `for`

```
l = [1,2,3,4,5]
for e in l:
  print(e)
```

- Nous pouvons utiliser la notation « tranche » pour obtenir des sous-listes (ou un élément individuel) : `l[i:j:k]` donne la sous-liste contenant tous les `k` éléments dont l'index est compris entre `i` et `j`.

```
l = [1,2,3,4,5,6,7,9,10]
print(l[0]) #?
print(l[1:]) #?
print(l[:1]) #?
print(l[-1]) #?
print(l[3:5]) #?
print(l[5:3]) #?
print(l[1:-2:1]) #?
print(l[1:-2:2]) #?
print(l[::2]) #?
```

Les listes sont modifiables

- De nombreuses méthodes importantes opérant sur les listes ne renvoient rien : elles modifient plutôt la liste elle-même.
- Les fonctions définies par l'utilisateur peuvent également modifier les listes :

```
def myAppend(l,x):
  l.append(x)
  return "Élément ajouté!"

l = []
myAppend(l,0)
print(l)
```

Tuples

- Les tuples sont, comme les listes, des types de données séquentiels.
- Une grande différence : ils sont **immuables** !
- Généralement utilisés pour stocker une séquence de données hétérogènes (contrairement aux listes).
- Elements accessibles via du « unpacking » ou via les indices.

```
t = ("key",0)

k, v = t
print(k == t[0])
print(v == t[1])
```

```
def f(t):
  t[0] = 1
  t[1] = 2
  return t

print(t == f(t))
```