TD 2 - Listes

Ferdinand Mom

April 2019

Figure 1: Meme de qualité

1 Classiques

Exercice 1.1: sumL

Créez la fonction **sumL(L)** qui retourne la somme des éléments de L.

```
res = sumL([])
print(res)
>>> 0

res = sumL([1, 2, 3])
print(res)
>>> 6
```

Exercice 1.2: average

Créez la fonction **average(L)** qui retourne la moyenne des élements de L. Levez une exception lorsque la liste est vide.

```
res = average([])
print(res)
>>> List is empty. Cannot divide by 0.

res = average([1, 2, 3])
print(res)
>>> 2
```

Exercice 1.3: listMult

Créez la fonction **listMult(L, x)** qui retourne une liste constituée d'éléments de L multipliés par x.

```
res = listMult([], 1)
print(res)
>>> []

res = listMult([1, 2, 3], 3)
print(res)
>>> [3, 6, 9]
```

Exercice 1.4: listDiv

Créez la fonction **listDiv(L, x)** qui retourne une liste constituée d'éléments de L divisés par x.

```
res = listDiv([], 1)
print(res)
>>> []
```

```
res = listDiv([1, 2, 3], 3)
print(res)
>>> [0.3333333333333333, 0.6666666666666666, 1.0]
```

Exercice 1.5: count

Créez la fonction **count(x, L)** qui retourne le nombre d'occurrences de x dans L.

```
res = count(1, [])
print(res)
>>> 0

res = count(3, [1, 2, 3, 3])
print(res)
>>> 2
```

Exercice 1.6: search

Créez la fonction **search(x, L)** qui retourne *True* si x est dans L.

```
res = search(1, [])
print(res)
>>> False

res = search(3, [1, 2, 3])
print(res)
>>> True
```

Exercice 1.7: nth

Créez la fonction **nth(index, L)** qui retourne l'élément contenu à l'index *index* de L. Levez une exception lorsque *index* est négatif ou lorsque la liste est trop courte.

```
res = nth(1, [])
print(res)
>>> "Input index is negative or list is too short."

res = nth(2, [1, 2, 3])
print(res)
>>> 3
```

Exercice 1.8: maximum

Créez la fonction **maximum(L)** qui retourne le plus grand élément de L. Levez une exception lorsque la liste est vide.

```
res = maximum([])
print(res)
>>> "List is empty."

res = maximum([1, 2, 3])
print(res)
>>> 3
```

Exercice 1.9: arithList

Créez la fonction **arithList(n, a1, r)** qui retourne une liste constituée des n termes d'une suite arithmétique de premier terme $a1$ et de raison r.

```
res = arithList(12, 2, 1)
print(res)
>>> [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

res = arithList(12, 2, -1)
print(res)
>>> [2, 1, 0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

Exercice 1.10: concat

Créez la fonction **concat(L1, L2)** qui retourne une liste correspondant à la concaténation de L1 et L2.

```
res = concat([], [])
print(res)
>>> []

res = concat([1, 2], [3, 4, 5, 6])
print(res)
>>> [1, 2, 3, 4, 5, 6]
```

Exercice 1.11: incOrd

Créez la fonction **incOrd(L)** qui retourne *True* si L est triée dans l'ordre croissant

```
res = incOrd([])
print(res)
>>> True

res = incOrd([1, 2, 3, 4, 5, 6])
print(res)
>>> True
```

```
res = incOrd([1, 2, 4, 3])
print(res)
>>> False
```

Exercice 1.12: searchIncOrd

Créez la fonction **searchIncOrd(x, L)** qui retourne *True* si x appartient à la liste triée dans l'ordre croissant L.

```
res = searchIncOrd(1, [])
print(res)
>>> False

res = incOrd(4, [1, 2, 3, 4, 5, 6])
print(res)
>>> True

res = searchIncOrd(3, [1, 2, 4])
print(res)
>>> False
```

Exercice 1.13: removeFirstOcc

Créez la fonction **removeFirstOcc(x, L)** qui retourne une liste sans la première occurrence de x de L.

```
res = removeFirstOcc(1, [])
print(res)
>>> []

res = removeFirstOcc(1, [2, 3, 4, 5, 6])
print(res)
>>> [2, 3, 4, 5, 6]

res = searchIncOrd(1, [1, 1, 2, 4])
print(res)
>>> [1, 2, 4]
```

Exercice 1.14: insert

Créez la fonction **insert(x, L)** qui retourne une liste avec x ajouté dans la liste triée dans ordre croissant L.

```
res = insert(1, [])
print(res)
>>> [1]

res = insert(1, [2, 3, 4, 5, 6])
print(res)
>>> [1, 2, 3, 4, 5, 6]
```

Exercice 1.15: reverse

Créez la fonction **reverse(L)** qui retourne la liste L dans le sens inverse.

```
res = reverse([])
print(res)
>>> []

res = reverse([1, 2, 3, 4, 5, 6])
print(res)
>>> [6, 5, 4, 3, 2, 1]
```

Exercice 1.16: jonction

Créez la fonction **jonction(L, link)** qui retourne une *string* constituée d'éléments de L séparée par *link*.

```
res = jonction([], "o")
print(res)
>>> ""

res = jonction(["1", "2", "3"], "-")
print(res)
>>> "1-2-3"
```

Exercice 1.17: separation

Créez la fonction **separation(s, link)** qui retourne une liste constituée d'éléments de la *string s* sans *link*.

```
res = separation("", "-")
print(res)
>>> []

res = separation("1-2-3", "-")
print(res)
>>> ["1", "2", "3"]
```

Exercice 1.18: frequencyOfMax

Créez la fonction **frequencyOfMax(L)** qui retourne un *tuple* contenant le plus grand élément de la liste L et sa fréquence. Levez une exception lorsque la liste est vide.

```
res = frequencyOfMax([])
print(res)
>>> "List is empty."

res = frequencyOfMax([1, 5, 3, 5, 4])
print(res)
>>> (5, 2)
```

Exercice 1.19: intersection

Créez la fonction **intersection(L1, L2)** qui retourne une liste constituée des éléments que *L1* et *L2* ont en commun. *L1* et *L2* sont triées dans l'ordre croissant.

```
res = intersection([], [])
print(res)
>>> []

res = intersection([1, 2, 3, 4, 5], [2, 5])
print(res)
>>> [2, 5]
```

Exercice 1.20: twistedList

Créez la fonction **twistedList(L1, L2)** qui retourne une liste constituée d'éléments de *L1* et *L2*. Le premier élément sera celui de *L1*, le deuxième celui de *L2*, le troisième celui de *L1* etc ...

```
res = twistedList([], [])
print(res)
>>> []

res = twistedList([1, 3, 5], [2, 4, 6])
print(res)
>>> [1, 2, 3, 4, 5, 6]
```

Exercice 1.21: merge

Créez la fonction **merge(L1, L2)** qui retourne les listes *L1* et *L2* fusionées dans l'ordre croissant. Ces 2 dernières sont également triées dans l'ordre croissant.

```
res = merge([], [])
print(res)
>>> []

res = twistedList([2, 3, 6], [1, 4, 5])
print(res)
>>> [1, 2, 3, 4, 5, 6]
```

Exercice 1.22: bubbleSort

Créez la fonction **bubbleSort(L1)** qui retourne une liste triée dans ordre croissant en utilisant la méthode "**bubble sort**".
(https://en.wikipedia.org/wiki/Bubble_sort#/media/File:Bubble-sort-example-300px.gif.

```
res = bubbleSort([])
print(res)
>>> []

res = bubbleSort([5, 3, 1, 2, 6, 4])
print(res)
>>> [1, 2, 3, 4, 5, 6]
```

True knowledge exists in knowing that you know nothing.

- Socrates