Séance 2: Itérations et nombres approchés

L1 – Université Nice Sophia Antipolis

Objectifs:

- Fonctions définies par récurrence
- Premiers pas avec les boucles while
- Notion d'accumulateur

Exercice 1 (Factorielle, ⋆)
Écrivez une fonction `fact(n)` *qui renvoie* $n!$*, la factorielle de* n *:*

1. *en utilisant une récurrence ;*
2. *en utilisant une boucle while et un accumulateur.*

□

Exercice 2 (Factorielles, ⋆)

1. *Écrivez une fonction* `print_facts(n)` *qui affiche sur* n *lignes les entiers* $1!$, $2!$, $3!$, $\ldots$, $n!$.
2. *Si vous avez utilisez la fonction* `fact(n)` *précédente, combien de multiplications sont effectuées par votre code lorsque vous exécutez* `print_facts(n)` *?*
3. *Proposez une solution utilisant moins de multiplications.*

□

Exercice 3 (Logarithme entier, ⋆)

Soit n *un entier positif. On appele* logarithme entier *de* n *l'entier* $\mathsf{le}(n)$ *correspondant au nombre de fois où il faut diviser* n *par deux avant d'atteindre* 1 *ou* 0*. Par exemple,* $\mathsf{le}(5) = 1 + \mathsf{le}(2) = 1 + (1 + \mathsf{le}(1)) = 2$*. Écrivez une fonction* `le(n)` *qui renvoie le logarithme entier de* `n`. □

Exercice 4 (Partie entière de la racine carrée, ⋆)

Soit n *un entier positif. La partie entière de la racine carrée de* n*, notée* $\lfloor \sqrt{n} \rfloor$*, est le plus grand entier* k *tel que* $k^2 \leq n$*. Écrivez une fonction* `int_sqrt(n)` *qui renvoie la partie entière de la racine carrée de* `n`. □

Complément sur la fonction print

Par défaut, la fonction `print` affiche un caractère espace entre chacun de ses arguments et termine en allant à la ligne à la fin. Deux paramètres optionnels de la fonction `print` permettent de modifier ce comportement par défaut : `sep` et `end`. Par exemple, le programme

```
print('Comportement','par','defaut.',sep=' ',end='\n')
print('Sur','une','seule','ligne','avec','des','tirets',sep='-',end='')
print('.\n','Nouvelle ligne','.')
```

affiche

```
Comportement par defaut.
Sur-une-seule-ligne-avec-des-tirets.
 Nouvelle ligne .
```

Le caractère `'\n'` est un caractère de contrôle qui lorsqu'il est affiché provoque un passage à la ligne. Dans la deuxième instruction `print`, on a remplacé le caractère de fin `'\n'` par `''` : `print` ne va pas à la ligne.

Complément sur les chaînes de caractères

On peut convertir un entier en chaîne de caractères : `str(42) == '42'`. Réciproquement, on peut convertir une chaîne de caractères en entier : `int('42') == 42`. Pour concaténer deux chaînes de caractères, on utilise l'opérateur +. On a par exemple `'caram' + 'bolage' == 'carambolage'` ; ce $+$ n'a rien à voir avec l'addition sur les entiers : `'2' + '3'` vaut `'23'`, mais pas `'3' + '2'`.

Exercice 5 (Écriture binaire, ⋆⋆– ⋆⋆⋆)

1. *Écrivez une fonction* `affiche_calcul_binaire(n)` *qui affiche le calcul de la représentation binaire de* `n`*, de sorte que l'on peut lire* verticalement *la représentation en binaire de* n*. Par exemple, pour* $13 = (1101)_2$*, on aura au toplevel*

```
>>> affiche_calcul_binaire(13)
1 ( 13 = 1 + 2 * 6 )
0 ( 6 = 0 + 2 * 3 )
1 ( 3 = 1 + 2 * 1 )
1
```

2. *Écrivez une fonction récursive* `affiche_binaire(n)` *qui affiche l'écriture binaire de* n *dans le sens de lecture usuel. Par exemple,* `affiche_binaire(13)` *affiche* `1101`.
3. *Écrivez cette fonction en utilisant une boucle* `while` *et un accumulateur.*

□

Exercice 6 (Décomposition en facteurs premiers, ⋆⋆⋆)

Écrivez une fonction `affiche_decomp(n)` *qui affiche le calcul de la décomposition en facteurs premiers de* `n`*. Par exemple,* `affiche_decomp(1176)` *affichera*

```
1176 = 2 ** 3 * 147
147 = 3 ** 1 * 49
49 = 7 ** 2
```

□