Bases de la programmation impérative (BPI)

CM4 - Plein de choses intéressantes dont des fonctions récursives

Manuel Selva

Sommaire du jour

Précisions concernant l'examen

Rappel TA vs SDD

Un problème : fonctions récursives

Questionnaire

(Un quiz : massifs environnants)

Précisions concernant l'examen

- 1h30 sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours
- exercice portant sur le projet

Précisions concernant l'examen

- 1h30 sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours
- exercice portant sur le projet

ATTENTION

- tester son programme au fur et à mesure

Précisions concernant l'examen

- 1h30 sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours
- exercice portant sur le projet

ATTENTION

- tester son programme au fur et à mesure
- `l=[]; i=0; while True: l.append[i]; i+=1`

Précisions concernant l'examen

- 1h30 sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours
- exercice portant sur le projet

ATTENTION

- tester son programme au fur et à mesure
- `l=[]; i=0; while True: l.append[i]; i+=1`
- éditer le fichier seulement avec `vscode`

Précisions concernant l'examen

- 1h30 sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours
- exercice portant sur le projet

ATTENTION

- tester son programme au fur et à mesure
- `l=[]; i=0; while True: l.append[i]; i+=1`
- éditer le fichier seulement avec `vscode`
- `ctrl + s` (ou `File`→`Save`) dans `vscode`

Précisions concernant l'examen

- 1h30 sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours
- exercice portant sur le projet

ATTENTION

- tester son programme au fur et à mesure
- `l=[]; i=0; while True: l.append[i]; i+=1`
- éditer le fichier seulement avec `vscode`
- `ctrl + s` (ou `File`→`Save`) dans `vscode`
- `envoyer/send` toutes les 15 minutes

Précisions concernant l'examen

- 1h30 sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours
- exercice portant sur le projet

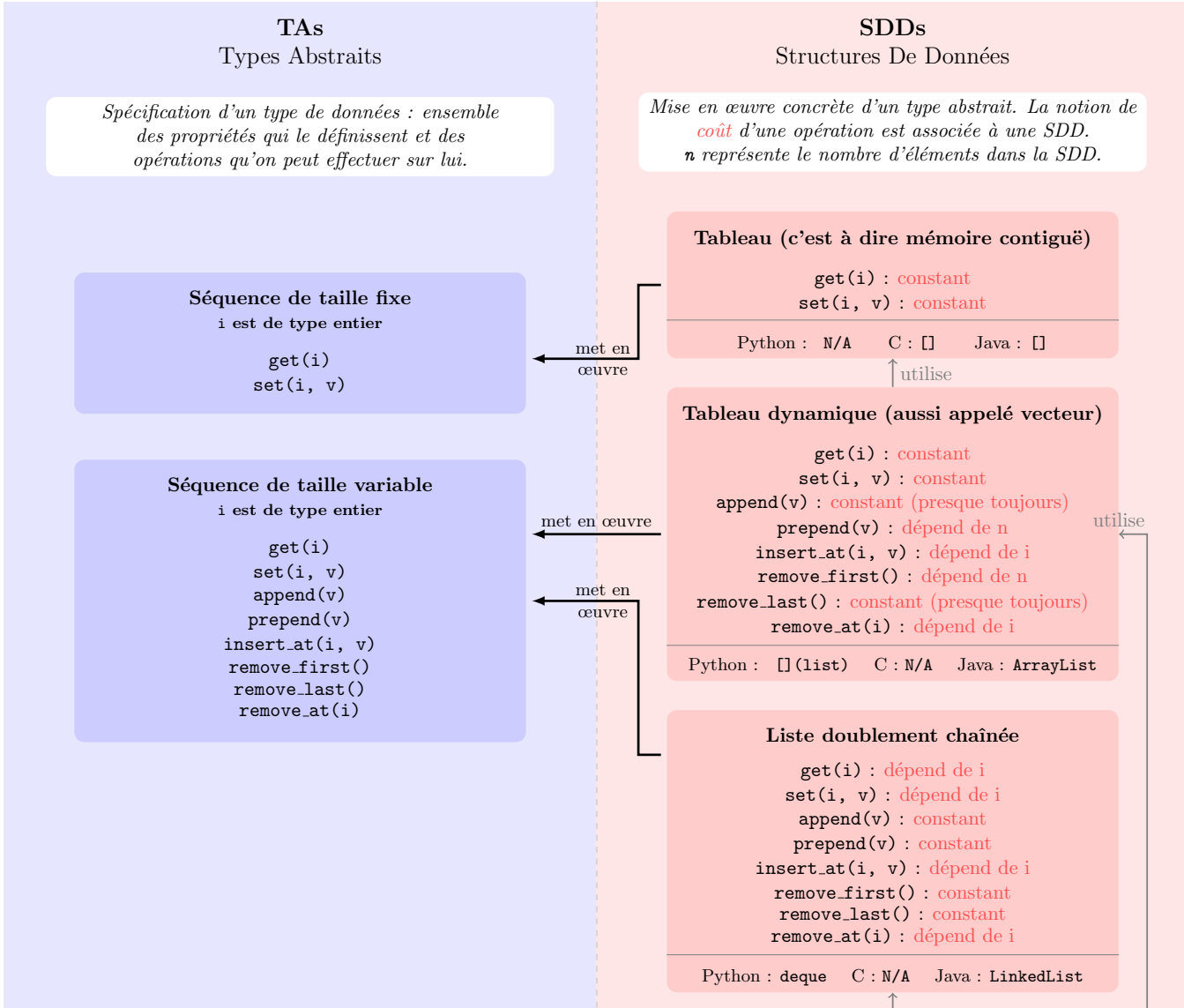
ATTENTION

- tester son programme au fur et à mesure
- `l=[]; i=0; while True: l.append[i]; i+=1`
- éditer le fichier seulement avec `vscode`
- `ctrl + s` (ou `File`→`Save`) dans `vscode`
- `envoyer/send` toutes les 15 minutes
- `envoyer et finir / send and exit` à la fin seulement ;-)

`list` vs liste chaînée vs tableau dynamique

`list` vs liste chaînée vs tableau dynamique

`list` == tableau dynamique
tableau dynamique != liste chaînée



Sommaire du jour

Précisions concernant l'examen

Rappel TA vs SDD

Un problème : fonctions récursives

Questionnaire

(Un quiz : massifs environnants)

Qu'est-ce que la récursivité ?

Qu'est-ce que la récursivité ?

Allons voir dans le dictionnaire 2013, Le Petit Larousse Illustré

- récursivité : propriété de ce qui est récursif
- récursif : qui peut être répété de façon indéfinie
- indéfini·e : que l'on ne peut délimiter ; infini·e

Qu'est-ce que la récursivité ?

Allons voir dans le dictionnaire 2013, Le Petit Larousse Illustré

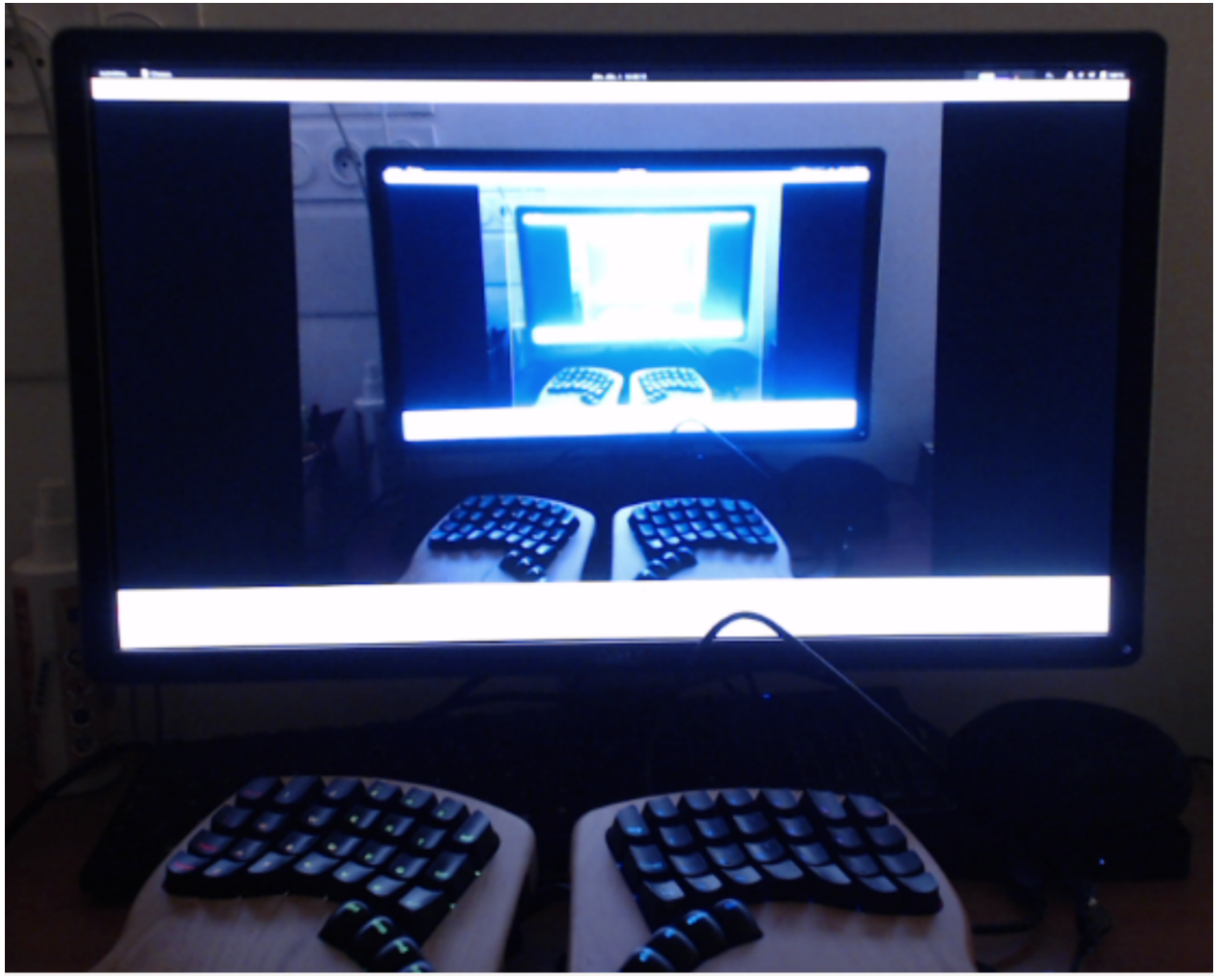
- récursivité : propriété de ce qui est récursif
- récursif : qui peut être répété de façon indéfinie
- indéfini·e : que l'on ne peut délimiter ; infini·e

Qu'est-ce que la récursivité ?

Rentrons dans la matrice

Qu'est-ce que la récursivité ?

Rentrons dans la matrice



Qu'est-ce qu'une fonction récursive ?

- Jusqu'à maintenant, on a résolu des problèmes avec des boucles, on parle de solutions **itératives**.

Qu'est-ce qu'une fonction récursive ?

- Jusqu'à maintenant, on a résolu des problèmes avec des boucles, on parle de solutions **itératives**.

- Certains problèmes se résolvent de façon **élégante** en résolvant des sous problèmes de même nature, par exemple :
  - comment dessiner un choux romanesco (pourquoi pas) ?
  - comment récupérer toutes les k-combinaisons d'une séquence ?

Qu'est-ce qu'une fonction récursive ?

- Jusqu'à maintenant, on a résolu des problèmes avec des boucles, on parle de solutions **itératives**.

- Certains problèmes se résolvent de façon **élégante** en résolvant des sous problèmes de même nature, par exemple :
  - comment dessiner un choux romanesco (pourquoi pas) ?
  - comment récupérer toutes les k-combinaisons d'une séquence ?

- On utilise dans ce cas là des fonctions qui s'appellent elles mêmes, dites **fonctions récursives**.

Qu'est-ce qu'une fonction récursive ?

- Jusqu'à maintenant, on a résolu des problèmes avec des boucles, on parle de solutions **itératives**.

- Certains problèmes se résolvent de façon **élégante** en résolvant des sous problèmes de même nature, par exemple :
  - comment dessiner un choux romanesco (pourquoi pas) ?
  - comment récupérer toutes les k-combinaisons d'une séquence ?

- On utilise dans ce cas là des fonctions qui s'appellent elles mêmes, dites **fonctions récursives**.

Pour bien comprendre le flot de contrôle d'une fonction récursive, nous allons regarder, à des fins pédagogiques uniquement, une fonction récursive qui n'est pas du tout élégante.

Organisation : comme d'habitude

- 5mn pour résoudre **individuellement**
- 2mn pour échanger avec **un seul** voisin
- vote
- Xmn pour débattre
- vote
- Xmn pour débattre à nouveau
- Xmn pour comprendre la bonne réponse

Que fait le programme ci-dessous ?

```
1  def m(i, elems):
2      idx = len(elems) - 1 - i
3      elems[idx] += 1
4      if i == 0:
5          return 1
6      return m(i - 1, elems) + \
7              m(i - 1, elems)
8
9  i = 5
10 elems = [0] * (i + 1)
11 print(m(i, elems), elems)
12 i = 60
13 elems = [0] * (i + 1)
14 print(m(i, elems), elems)
```

A. affiche deux lignes avec plusieurs 1 sur chacune des deux lignes ;

B. affiche deux lignes avec plusieurs puissances de 2 sur chacune des 2 lignes ;

C. autre ;

D. je ne sais pas.

Exécutons ce programme pas à pas

control

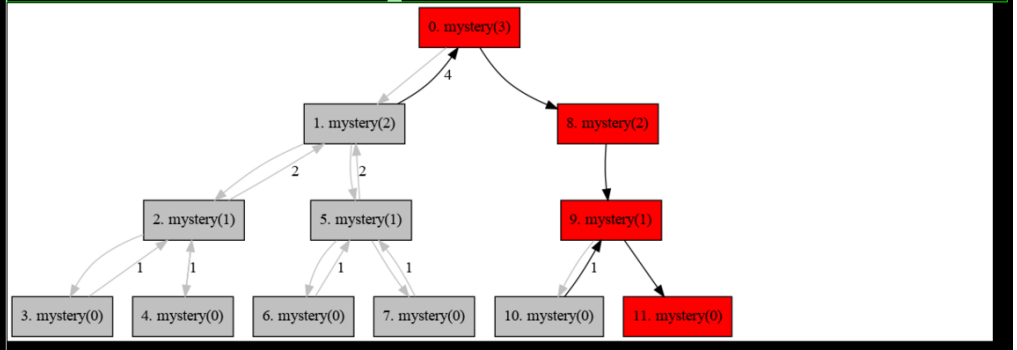
```
 1 #!/usr/bin/env python3
 2
 3 def mystery(x, y):
 4     z = x
 5     a = len(y) - 1 - x
 6     y[a] += 1
 7     if x == 0:
 8         return 1
 9     m1 = mystery(z - 1, y)
10     m2 = mystery(x - 1, y)
11     return m1 + m2
12
13
14 def main():
15     r = 3
16     b = [0] * r
17     print(mystery(r, b), b)
18
19 if __name__ == "__main__":
20     main()
~
~
~
~
~
~
~
~
~
~
"mystery.py" 20L, 302B
```

```
Taper entrée pour avancer :
Taper entrée pour avancer :
Taper entrée pour avancer :
Taper entrée pour avancer :

Taper entrée pour avancer :
Taper entrée pour avancer :

Taper entrée pour avancer :

Taper entrée pour avancer :Taper entrée pour avancer :
Taper entrée pour avancer :
Taper entrée pour avancer :
Taper entrée pour avancer :
```



./tester_all.py | manu@casper:~/boulot/teach/git-bpi-pro… | control

Des combinaisons élégantes !

```python
def recupere_combinaisons_rec(sequence, k):
    """Version récursive, élégante, non ?"""

    def combinaisons_aux(start, k, prefix):
        """Cache les paramètres "internes" nécessaires pour la recursion."""

        # Cas de base : on a ajouté k éléments
        # dans prefix, c'est une k-combinaisons
        if k == 0:
            combinaisons.append(tuple(prefix))

        # Sinon, pour chaque élément d'indice supérieur ou égale à start,
        # on le rajoute, puis on fait un appel récursif.
        else:
            for indice in range(start, len(sequence)):
                prefix.append(sequence[indice])
                combinaisons_aux(indice + 1, k - 1, prefix)
                prefix.pop()

    # On initialise, on remplit et on renvoie
    # la liste contenant les k-combinaisons.
    combinaisons = []
    combinaisons_aux(0, k, [])
    return combinaisons
```

À retenir

- certains problèmes se résolvent naturellement avec des fonctions récursives

À retenir

- certains problèmes se résolvent naturellement avec des fonctions récursives
- une fonction récursive doit toujours avoir un cas de base

À retenir

- certains problèmes se résolvent naturellement avec des fonctions récursives
- une fonction récursive doit toujours avoir un cas de base
- la "taille du problème" doit diminuer à chaque appel pour arriver au cas de base au bout d'un certain nombre d'appels récursifs
- il faut maîtriser le flot de contrôle de ses fonctions récursives (dessiner l'arbre d'appels aide grandement)

À retenir

- certains problèmes se résolvent naturellement avec des fonctions récursives
- une fonction récursive doit toujours avoir un cas de base
- la "taille du problème" doit diminuer à chaque appel pour arriver au cas de base au bout d'un certain nombre d'appels récursifs
- il faut maîtriser le flot de contrôle de ses fonctions récursives (dessiner l'arbre d'appels aide grandement)
- la complexité peut se déduire en comptant les nœuds de l'arbre d'appels

Questionnaire

Merci pour vos retours nombreux et constructifs :)

Sommaire du jour

Précisions concernant l'examen

Rappel TA vs SDD

Un problème : fonctions récursives

Questionnaire

(Un quiz : massifs environnants)

Quel est le nom de ce massif ?

Quel est le nom de ce massif ?

Quel est le nom de ce massif ?

Quel est le nom de ce massif ?