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

Fonctions récursives

(Un quiz : massifs environnants)

Précisions concernant l'examen

- 2h sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours

Précisions concernant l'examen

- 2h sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours

ATTENTION

- tester son programme au fur et à mesure

Précisions concernant l'examen

- 2h sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours

ATTENTION

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

Précisions concernant l'examen

- 2h sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours

ATTENTION

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

Précisions concernant l'examen

- 2h sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours

ATTENTION

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

Précisions concernant l'examen

- 2h sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours

ATTENTION

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

Précisions concernant l'examen

- 2h sur machine de l'école
- environnement examen
  - répertoire personnel vide
  - pas d'internet
  - accès au site du cours

ATTENTION

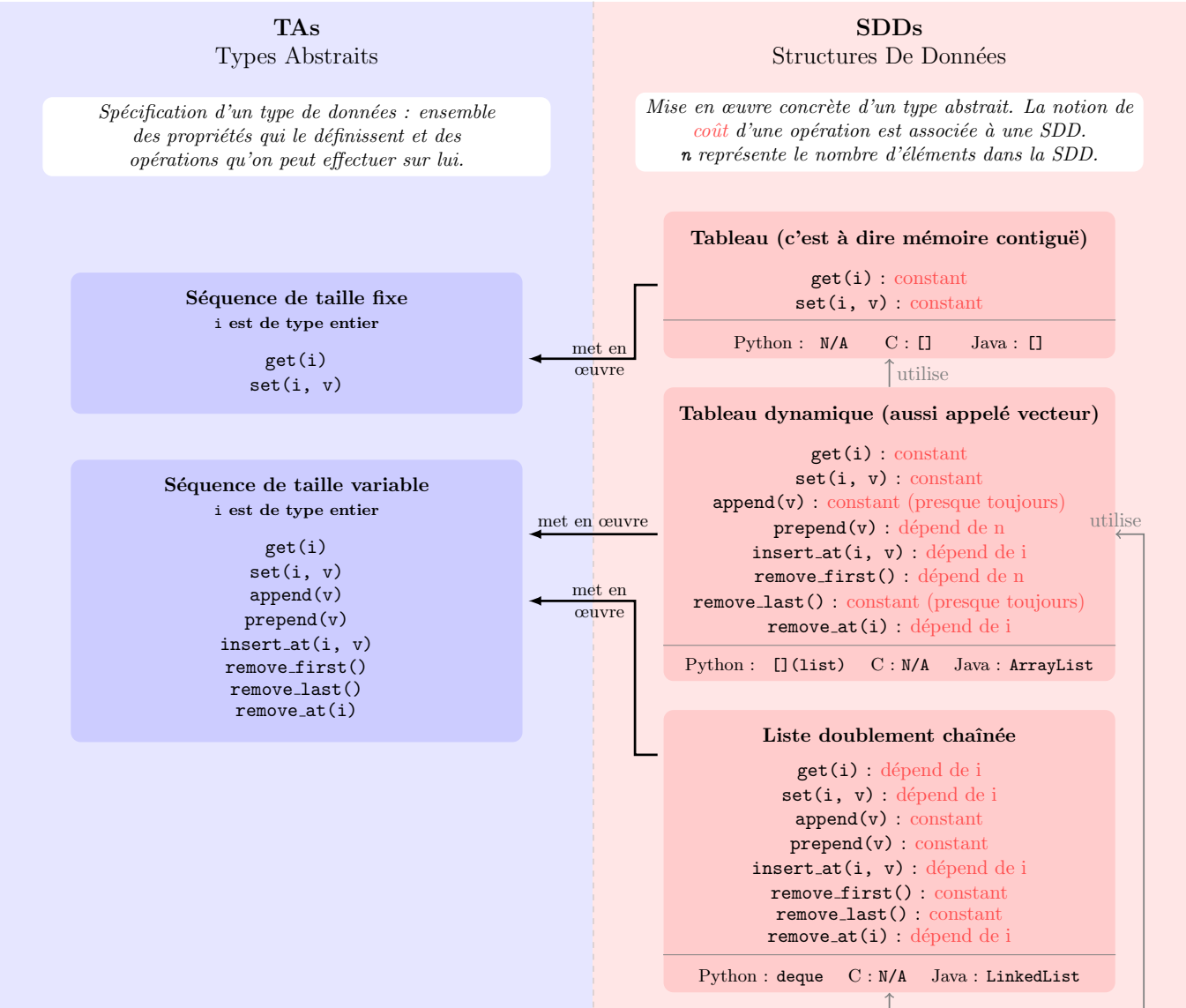
- tester son programme au fur et à mesure
- `l=[]; i=0; while True: l.append[i]; i+=1`
- éditer vos fichiers seulement avec `vscode`
- `ctrl + s` (ou `File`→`Save`) dans `vscode`
- `envoyer/send` toutes les 15 minutes
- `envoyer et finir / send and exit` à la fin seulement ;-)

`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

Fonctions récursives

(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

Qu'est-ce que la récursivité ?

Allons voir dans le dictionnaire 2013, Le Petit Larousse Illustré

- récursivité : propriété de ce qui est récursif

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

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

Allons voir sur wikipedia

La récursivité est une démarche qui fait référence à l'objet même de la démarche à un moment du processus. En d'autres termes, c'est une démarche dont la description mène à la répétition d'une même règle.

Dans les maths

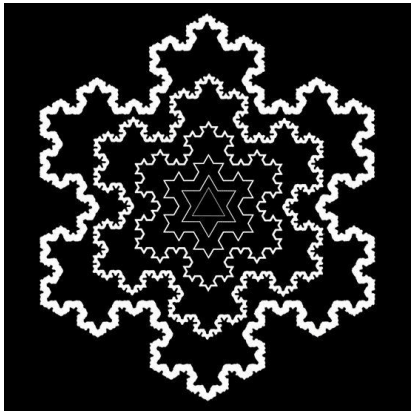
- Ensembles définis récursivement

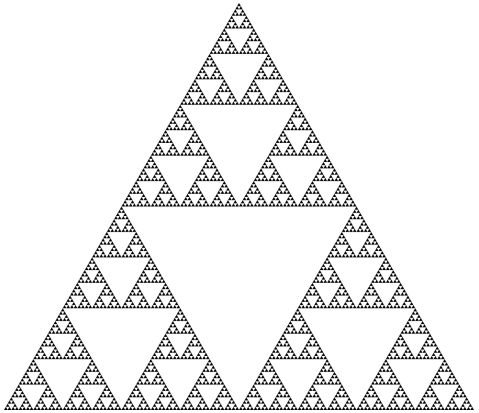
Dans les maths

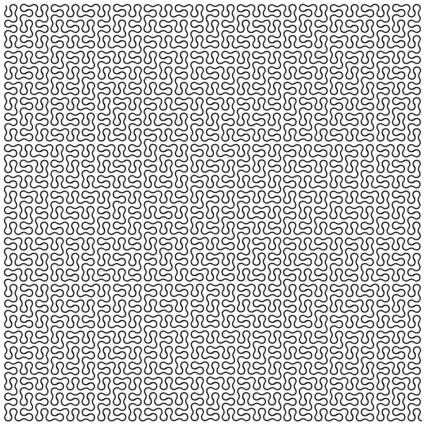
- Ensembles définis récursivement
- Suites définies par récurrence
  - Fibonacci
  - Syracuse

Dans les maths

- Ensembles définis récursivement
- Suites définies par récurrence
  - Fibonacci
  - Syracuse
- Fractales : objets mathématiques avec structure similaire à toutes les échelles

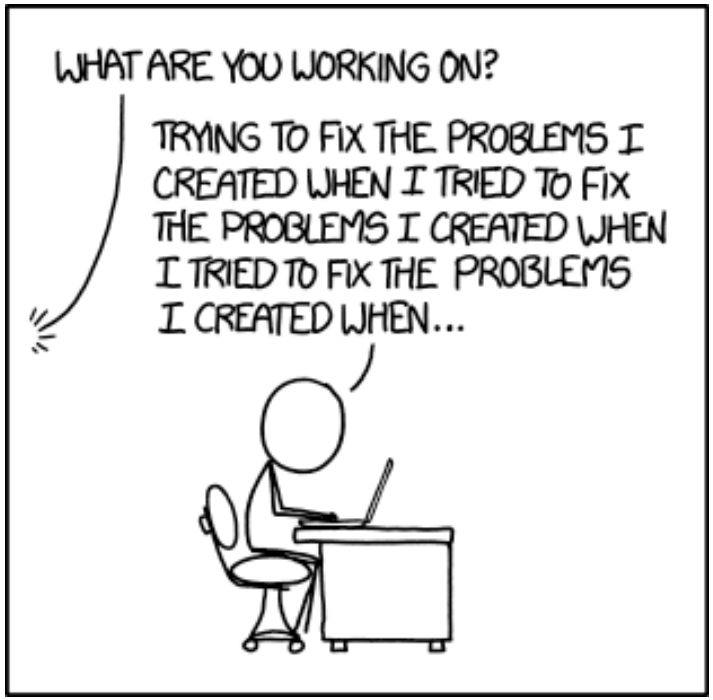


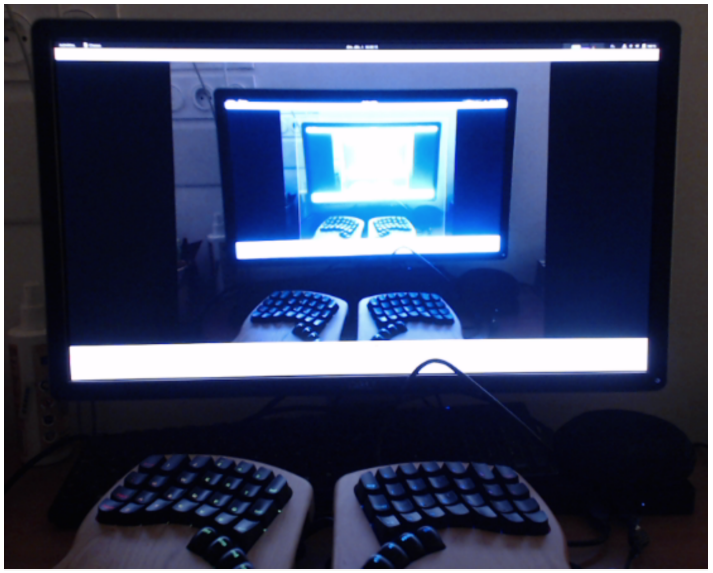


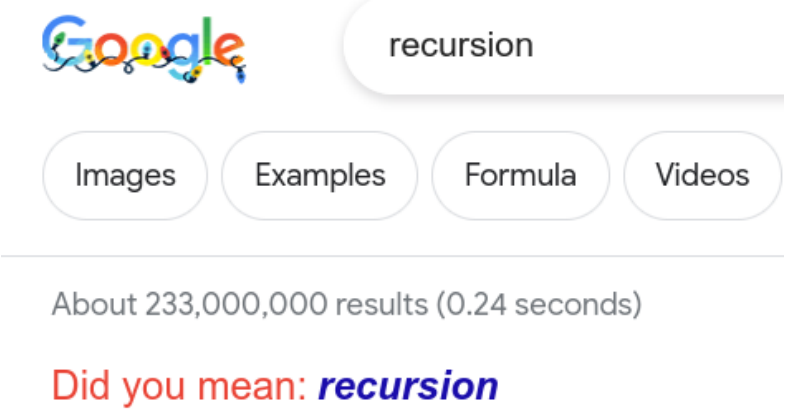


Dans la nature

Dans l'humour (enfin, une certaine forme !)







GNU : GNU is not Unix

Et en informatique ?

Et en informatique ?

Fonctions récursives

- Dans les chapitres 1, 2 et 3 on a résolu tous nos problèmes avec des boucles, on parle de solutions **itératives**.

Et en informatique ?

Fonctions récursives

- Dans les chapitres 1, 2 et 3 on a résolu tous nos problèmes avec des boucles, on parle de solutions **itératives**.
- Certains problèmes se résolvent de façon **élégante** en résolvant des sous problèmes de même nature

Et en informatique ?

Fonctions récursives

- Dans les chapitres 1, 2 et 3 on a résolu tous nos problèmes avec des boucles, on parle de solutions **itératives**.
- Certains problèmes se résolvent de façon **élégante** en résolvant des sous problèmes de même nature
- On utilise dans ce cas là des fonctions qui s'appellent elles mêmes, dites **fonctions récursives**.

Et en informatique ?

Fonctions récursives

- Dans les chapitres 1, 2 et 3 on a résolu tous nos problèmes avec des boucles, on parle de solutions **itératives**.
- Certains problèmes se résolvent de façon **élégante** en résolvant des sous problèmes de même nature
- On utilise dans ce cas là des fonctions qui s'appellent elles mêmes, dites **fonctions récursives**.

Structures récursives

- Listes chaînées
- Arbres

En entretien d'embauche on vous demande que fait le programme ci-dessous ?

```
1  def m(i):
2      if i == 0:
3          return 1
4      return m(i - 1) + m(i - 1)
5
6
7  print(m(3))
8  print(m(60))
```

Des propositions

- Affiche 8 et 1152921504606846976
- Génère une erreur mémoire
- Autre

En CM de BPI on vous demande que vaut la variable a dans le programme ci-dessous ?

```
1  def g(a):
2      if a == 0:
3          a = 32
4          g(1)
5      elif a == 1:
6          a = 64
7          g(2)
8      else:
9          a = 1
10         return a
11
12 def main():
13     a = 0
14     a = g(a)
15
16 main()
```

Que fait cette fonction ?

```
1  def f_iter(sequence, k):
2      n = len(sequence)
3      indices = list(range(k))
4      res = []
5      res.append(tuple(sequence[i] for i in indices))
6      while True:
7          for i in reversed(range(k)):
8              if indices[i] != i + n - k:
9                  break
10         else:
11             break
12         indices[i] += 1
13         for j in range(i + 1, k):
14             indices[j] = indices[j - 1] + 1
15         res.append(tuple(sequence[i] for i in indices))
16     return res
```

Que fait cette fonction ?

```
1  def f_rec(sequence, k):
2      def f_aux(start, k, prefix):
3          if k == 0:
4              res.append(tuple(prefix))
5          else:
6              for indice in range(start, len(sequence)):
7                  prefix.append(sequence[indice])
8                  f_aux(indice + 1, k - 1, prefix)
9                  prefix.pop()
10
11     res = []
12     f_aux(0, k, [])
13     return res
```

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 le chronogramme d'exécution et/ou 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 le chronogramme d'exécution et/ou l'arbre d'appels aide grandement)
- la complexité peut se déduire en comptant les nœuds de l'arbre d'appels

Sommaire du jour

Précisions concernant l'examen

Rappel TA vs SDD

Fonctions récursives

(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 ?