

Base de programmation

- BA1 Informatique
Johan Depréter – johan.depreter@heh.be

Quiz

Correction

- Problème :
Comment savoir si le mot rentré est un palindrome

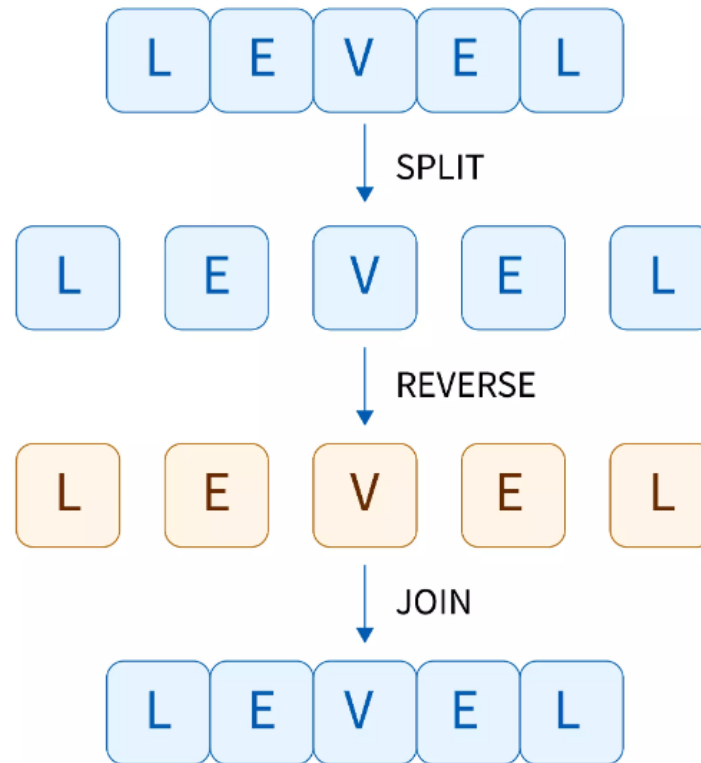
Entrée x – mot

Pré-condition – x est un mot

Sortie z – mot inversé

Post-condition – $z = 1 / x$

Correction



Correction

- Problème :
Comment savoir si le mot rentré est un palindrome

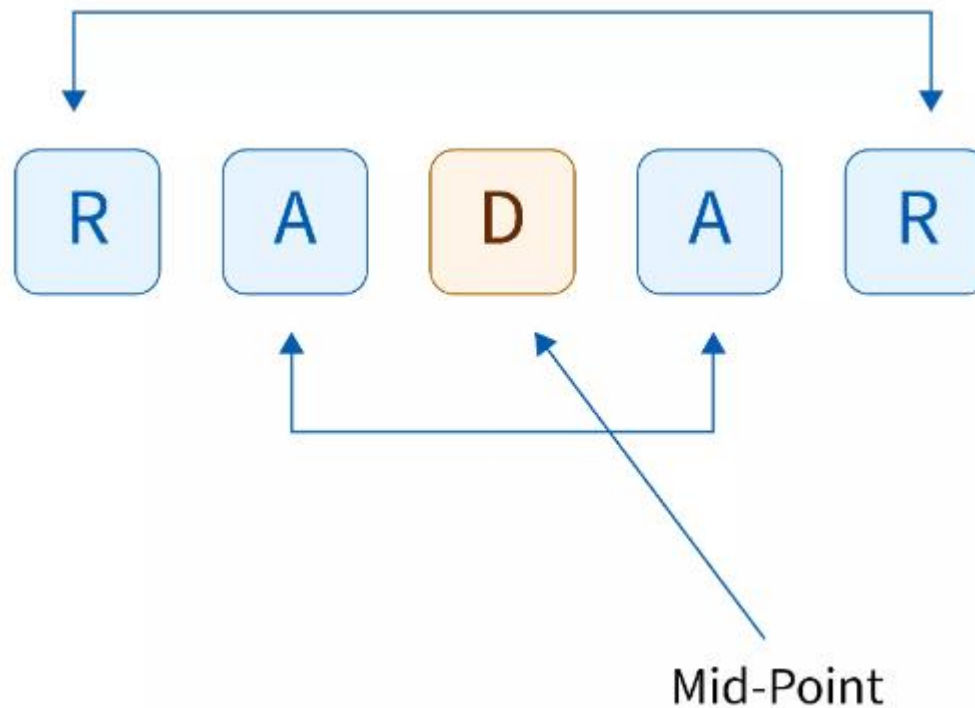
Entrée x – mot

Pré-condition – x est un mot

Sortie z – Vrai ou faux

Post-condition – $x = 1 / x$

Correction



- Problème :

Valeurs min et max calculées à partir d'une sélection de 4 parmi une liste de 5 entiers

Entrée x – liste

Pré-condition – liste contient des entiers

Sortie y – min

Sortie z – max

Post-condition – $\min = x_1 + x_2 + x_3 + x_4 < \sum x_i$
 $\max = x_1 + x_2 + x_3 + x_4 > \sum x_i$

Correction

```
entiers = [12, 5, 8, 21, 9]

# Calcul de la somme totale
somme_totale = sum(entiers)

# Valeur minimale : soustraction du plus grand entier
valeur_minimale = somme_totale - max(entiers)

# Valeur maximale : soustraction du plus petit entier
valeur_maximale = somme_totale - min(entiers)

print("Valeur minimale :", valeur_minimale)
print("Valeur maximale :", valeur_maximale)
```


Correction

```
entiers = [12, 5, 8, 21, 9]

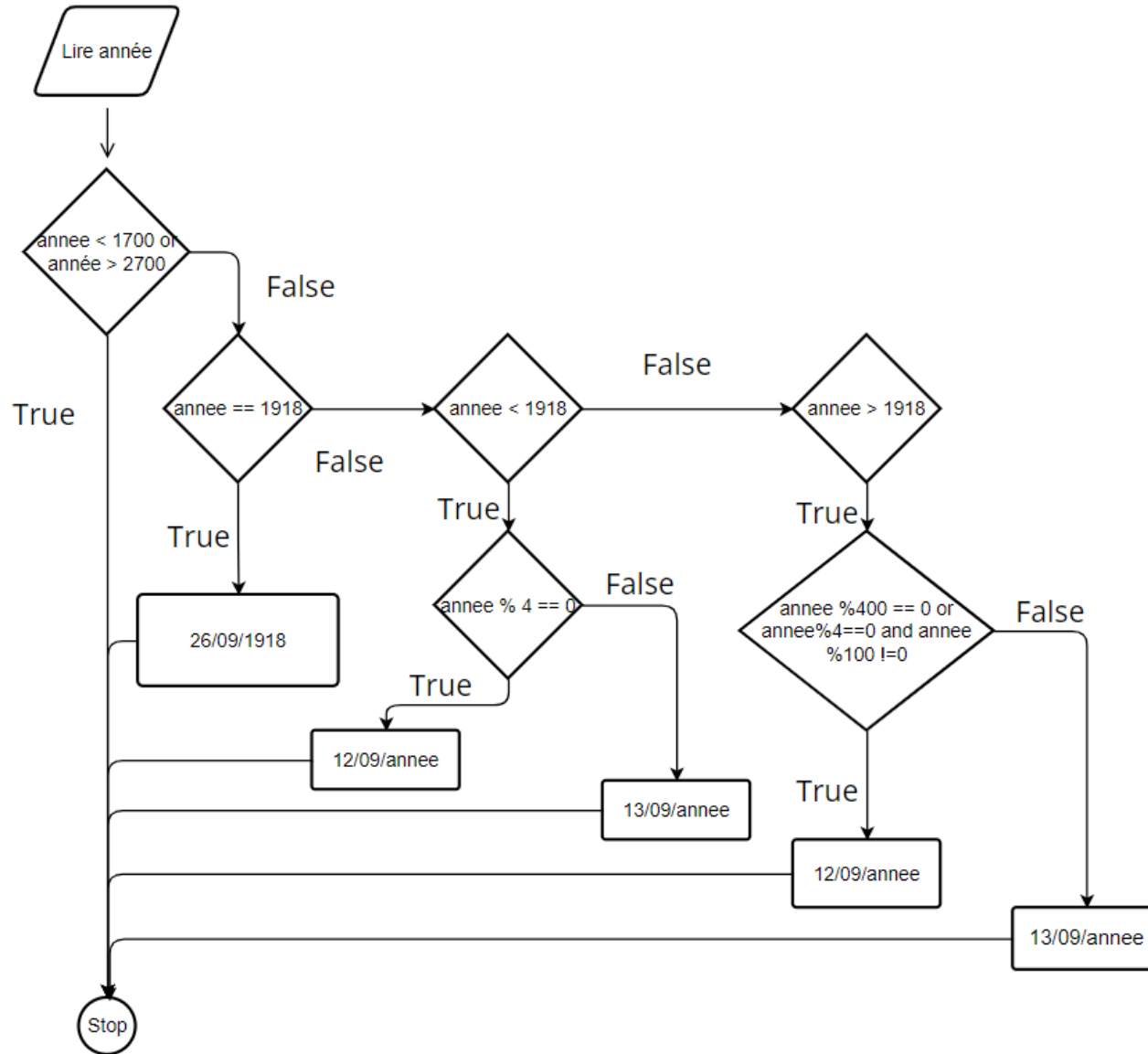
# Tri de la liste par ordre croissant
entiers.sort()

# Valeur minimale : somme des quatre plus petits entiers
valeur_minimale = sum(entiers[:4])

# Valeur maximale : somme des quatre plus grands entiers
valeur_maximale = sum(entiers[1:])

print("Valeur minimale :", valeur_minimale)
print("Valeur maximale :", valeur_maximale)
```

Correction



Tours de Hanoï

- Problème :

On a 7 disques de diamètres différents qui forment une tour. On souhaite déplacer ces disques vers une nouvelle tour en suivant les règles suivantes :

- On ne peut pas déplacer plus d'un disque à la fois
- On ne peut placer un disque que sur un disque plus grand (ou sur un emplacement vide)

Trouver comment résoudre ce problème avec le moins de déplacement possible, et en utilisant une tour intermédiaire.

Tours de Hanoï

- Rédiger les spécifications de la classe du problème
- Formaliser le problème

Chapitre 5

- **Les fonctions**

- Répétition de fonctionnalités
- Modularité
- Evolutivité

Notions de base

- Paramètres
- Variables locales
- Fonctions / Procédures
- Appels de fonctions

```
def add(a, b):  
    c = a + b  
    return c
```

```
somme = add(5, 2)
```

Paramètres

- Les paramètres obligatoires
- Les paramètres par défaut
- Les paramètres par mot-clé
- Les paramètres de taille variable

```
def add(a, b=5):  
    c = a + b  
    return c  
  
somme = add(5)
```

```
somme = add(b=5, a=2)
```

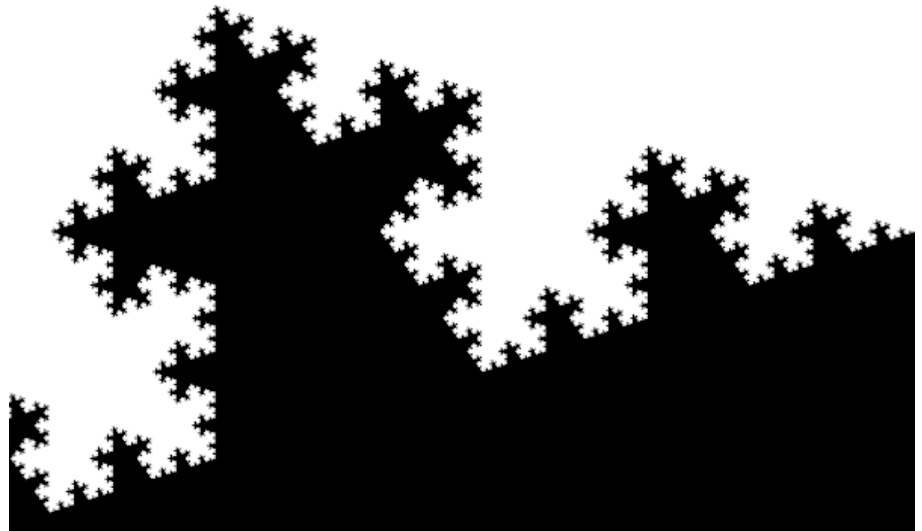
```
def add(a, *bs):  
    c = a  
    for b in bs:  
        c += b  
    return c  
  
somme = add(5, 2, 3)
```


Notions importantes

- Indépendance entre le programme et la fonction
- Transmission par valeur ou par référence
- Concordance des types

La récursivité

- La récursivité est une démarche qui fait référence à l'objet même de la démarche pendant son processus
- Exemples :
 - Le calcul d'une factorielle
 - La suite de fibonacci

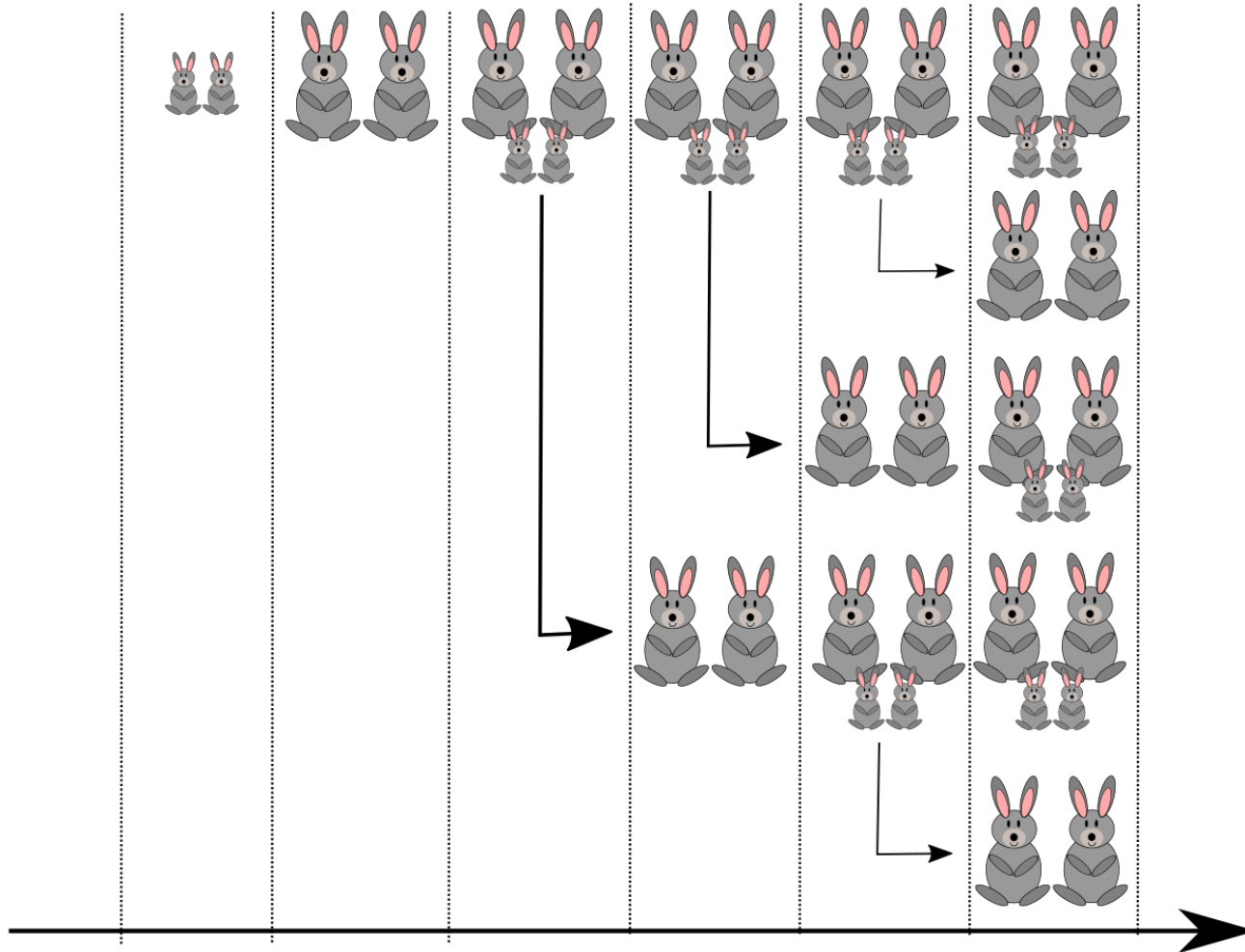


Problème des lapins

- Le problème posé est le suivant :

« Quelqu'un a déposé un couple de lapins dans un certain lieu, clos de toutes parts, pour savoir combien de couples seraient issus de cette paire en une année, car il est dans leur nature de générer un autre couple en un seul mois, et qu'ils enfantent dans le second mois après leur naissance. »

Problème des lapins



Problème des lapins

- Formulation de la classe du problème :
En sachant qu'un couple de lapins génère un nouveau couple de lapin chaque mois à partir de leur 2^{ème} mois d'existence, après x mois combien aurais-je de couple de lapins ?
- Spécifications du problème :
Entrée a : nombre de mois
Pré-condition : a réel positif
Sortie m : nombre de couples de lapins
Post-condition : ?

Suite de Fibonacci

0	1	2	3	4	5	6	7	8
0	1	1	2	3	5	8	13	21

- Cas général :
Un élément est égal à la somme des deux éléments qui le précèdent
- Cas de base :
L'élément 0 vaut 0 et l'élément 1 vaut 1

Suite de Fibonacci

```
def fibo(n):  
    if n == 0 or n == 1:  
        return n  
    else:  
        return fibo(n - 1) + fibo(n - 2)
```

- Pile d'exécution

