Christophe Viroulaud

Terminale - NSI

Lang 05

### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in"

Préconditions

M-----

Durée d'exécuti

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

Formulations

Notation mathématique

Implémentation Nouvelle formulation mathématique

Etude de la fonction native

Fonctions Python "built-in" Tester un programme

Préconditions

Mettre en place des te

Duree d executio

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

Formulations

lotation mathématique

nplémentation louvelle formulation nathématique

L'exponentiation est une opération mathématique définie par :

$$a^n = \underbrace{a \times .... \times a}_{n \text{ fois}}$$
 et  $a^0 = 1$ 

# $ightharpoonup 2^4 ightarrow 3$ opérations,

▶  $2701^{103056} \rightarrow 103055$  opérations.

Comment calculer la puissance d'un nombre de manière optimisée ?

### Exponentiation Notion de récursivité

Etude de la conction native

Fonctions Python "built-in"

Préconditions

Mari

Donda Pandanati

\_\_\_\_\_

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

ormulations écursives

Notation mathématique Implémentation Nouvelle formulation

### Exponentiation Notion de récursivité

### 1. Étude de la fonction native

- 1.1 Fonctions Python "built-in"
- 1.2 Tester un programme
- 2. Implémenter la fonction puissance
- 3. Formulations récursives

# Étude de la fonction native

Fonctions Python "built-in"

Préconditions

Mettre en place de

Durée d'exécution

Implémenter la

S'appuyer sur la définition mathématique

Correction de l'algorithme

\_ ...

Formulations

ecursives

otation mathematique

Nouvelle formulation mathématique

# Fonctions Python "built-in"

```
def puissance_star(x:int,n:int)->int:
    return x**n

def puissance_builtin(x:int,n:int)->int:
    return pow(x,n)
```

Code 1 - Fonctions natives

Activité 1 : Tester les deux fonctions du code 1.

#### Exponentiation Notion de récursivité

fonction native

### Fonctions Python "built-in"

Préconditions

Préconditions

Durée d'exécution

Implémenter la

nathématique

Correction de l'algorithme

Formulations récursives

Notation mathematique Implémentation Nouvelle formulation

### Exponentiation Notion de récursivité

### 1. Étude de la fonction native

- 1.1 Fonctions Python "built-in"
- 1.2 Tester un programme
  Préconditions
  Mettre en place des tests
  Durée d'exécution
- 2. Implémenter la fonction *puissance*
- Formulations récursives

Étude de la fonction native

Fonctions Python "built-in

Tester un programme

Préconditions

Mettre en place des te

Durée d'exécution

Implémenter la

mathématique

Correction de l'algorithme

Formulations

récursives

Notation mathematique Implémentation

Nouvelle formulation mathématique

### Préconditions

Nous décidons de nous limiter au cas positif.

# À retenir

La programmation *défensive* consiste à anticiper les problèmes éventuels.

Activité 2 : Mettre en place un test qui lèvera une AssertionError si l'exposant est négatif.

### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in" Tester un programme

#### Préconditions

iviettre en place di

Durée d'exécution

mplémenter la conction *puissance* 

mathématique Correction de l'algorithme

ormulations écursives

Notation mathématiqu

Implémentation
Nouvelle formulation

### Correction

3

```
def puissance_star(x: int, n: int) -> int:
    assert n >= 0, "L'exposant doit être positif."
    return x**n
```

Code 2

### Exponentiation Notion de récursivité

Etude de la conction native

Fonctions Python "built-in"

#### Préconditions

Mettre en place des tes

# Implémenter la fonction puissance

mathématique Correction de l'algorithme

Formulations

Notation mathématique Implémentation Nouvelle formulation

# Mettre en place des tests

Il existe plusieurs modules (doctest) qui facilitent les phases de test.

#### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

Formulations

Notation mathématique

Implémentation Nouvelle formulation mathématique

```
import doctest
1
2
    def puissance star(x:int,n:int)->int:
3
        11 11 11
4
        >>> puissance_star(2,8)
5
        256
6
        >>> puissance star(2,9)
        512
8
        11 11 11
9
        return x**n
10
11
    doctest.testmod(verbose=True)
12
```

Code 3 - Tester une fonction

Etude de la Conction native

Fonctions Python "built-in"

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la

mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique Implémentation

Nouvelle formulation mathématique

### Durée d'exécution

**Activité 3 :** À l'aide de la bibliothèque time mesurer la durée d'exécution de la fonction puissance\_star pour calculer  $2701^{19406}$ .

### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in' Tester un programme

Préconditions

#### Durée d'exécution

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique Implémentation Nouvelle formulation mathématique

### Correction

```
from time import time

debut=time()

puissance_star(2701,19406)

fin=time()

print("opérande **",fin-debut)
```

#### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in"

Préconditions

Durée d'evécution

#### Duree a executio

Implémenter la

mathématique

Correction de l'algorithme

Formulations

Notation mathématique

Implémentation Nouvelle formulation mathématique

### Exponentiation Notion de récursivité

1 Étude de la fonction native

- 2. Implémenter la fonction puissance
- 2.1 S'appuver sur la définition mathématique
- 2.2 Correction de l'algorithme
- 3 Formulations récursives

### Étude de la fonction native

Fonctions Python "built-in

Tester un programme Préconditions

Mettre en place des 1

Durée d'exécution

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

ormulations

écursives

lotation mathématique

mplémentation Vouvelle formulation

# S'appuyer sur la définition mathématique

$$a^n = \underbrace{a \times .... \times a}_{nfois}$$
 et  $a^0 = 1$ 

### Activité 4:

- Implémenter la fonction puissance\_perso(x : int, n : int) → int sans utiliser les fonctions buitin de Python.
- 2. Mettre en place un test de vérification de la fonction.
- 3. Mesurer le temps d'exécution de la fonction en l'appelant avec les paramètres (2701,19406).

### Exponentiation Notion de récursivité

Etude de la fonction native

Fonctions Python "built-in" Tester un programme

Préconditions Mettre en place des tests

mplémenter la

### S'appuyer sur la définition mathématique

Correction de l'algorithme

éCURSIVES Notation mathématique

Implémentation

Nouvelle formulation
mathématique

### Correction

```
def puissance perso(x:int,n:int)->int:
1
         11 11 11
 2
        >>> puissance perso(2,8)
 3
        256
 4
        >>> puissance perso(2,9)
5
        512
6
         11 11 11
8
        res = 1
9
        for i in range(n):
10
             res*=x
11
        return res
```

```
opérande ** 0.006058692932128906
fonction pow() 0.005688667297363281
fonction personnelle 0.13074541091918945
```

### Exponentiation Notion de récursivité

S'appuver sur la définition mathématique

Correction de l'algorithme

- 2. Implémenter la fonction puissance
- 2.1 S'appuyer sur la définition mathématique
- 2.2 Correction de l'algorithme
- 3 Formulations récursives

### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in"

Préconditions

Durée d'exécution

Implémenter la

S'appuyer sur la défir

mathématique

### Correction de l'algorithme

Formulations

OCUTSIVES

Implémentation

# Correction de l'algorithme

# À retenir

Un **invariant de boucle** est une propriété qui est vraie avant l'exécution de chaque itération.

### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in"

Préconditions

Durée d'exécutio

Implémenter la fonction *puissance* 

mathématique

### Correction de l'algorithme

ormulations écursives

Notation mathématique
Implémentation
Nouvelle formulation
mathématique

```
1 res = 1
2 for i in range(n):
```

Code 5 – La propriété  $res = x^i$  est un invariant de boucle.

3

res\*=x

#### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in" Tester un programme

Préconditions

Durée d'exécutio

Implémenter la

S'appuyer sur la défini

Correction de l'algorithme

ormulations

Notation mathématique Implémentation

Nouvelle formulation mathématique

▶ Si i = 0 (début de la première itération) la propriété est vérifiée :  $x^0 = 1 = res$ .

Etude de la fonction native

Fonctions Python "built-in"

Préconditions

Mettre en place des t

Durée d'exécution

mplémenter la fonction *puissanc*e

mathématique

Correction de l'algorithme

\_

cursives

lotation mathématique mplémentation

fonction native

Tester un programme

Préconditions Mettre en place des te

Durée d'exécutio

ightharpoonup Si i=0 (début de la première itération) la propriété est

vérifiée :  $x^0 = 1 = res$ .

Supposons la propriété vraie au rang p.

Implémenter la fonction puissance

S'appuyer sur la définiti mathématique

Correction de l'algorithme

ormulations

Notation mathématique Implémentation

Correction de l'algorithme

- Supposons la propriété vraie au rang p.
- Vérifions au rang p+1:
  - ightharpoonup au début de l'itération p, res =  $x^p$
  - ightharpoonup à la fin de l'itération p,  $res = x^p * x = x^{p+1}$
  - b donc au début de l'itération p+1,  $res = x^{p+1}$

- Exponentiation Notion de récursivité

- Correction de l'algorithme
- Formulations
- récursives

- 3. Formulations récursives

# Notation mathématique

$$\textit{puissance}(\textit{x},\textit{n}) = \left\{ \begin{array}{ll} 1 & \text{si } \textit{n} = 0 \\ \textit{x.puissance}(\textit{x},\textit{n} - 1) & \text{si } \textit{n} > 0 \end{array} \right.$$

# À retenir

Une fonction **récursive** est une fonction qui s'appelle ellemême.

#### Exponentiation Notion de récursivité

Étude de la fonction native

Tester un programme

Mettre en place des tests

Durée d'exécution

Implémenter la fonction *puissance* 

mathématique Correction de l'algorithme

Formulations

Notation mathématique

Implémentation Nouvelle formulation mathématique

- Exponentiation Notion de récursivité

- Correction de l'algorithme

- Implémentation

- 3. Formulations récursives
- 3.2 Implémentation

# Implémentation

# À retenir

Une fonction récursive :

- s'appelle elle-même,
- possède un cas limite pour stopper les appels.

### Exponentiation Notion de récursivité

Etude de la Gonction native

Fonctions Python "built-in"

Préconditions

D / II / I'

Durée d'exécuti

Implémenter la fonction *puissance* 

mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique

Implémentation

```
def puissance_recursif(x: int, n: int) -> int:
    if n == 0: # cas limite
    return 1
delse: # appel récursif
return x*puissance_recursif(x, n-1)
```

Code 6 - Traduction de la formule mathématique

Etude de la conction native

Fonctions Python "built-in" Tester un programme

Préconditions

Mettre en place u

fonction puissance

mathématique Correction de l'algorithme

Formulations

récursives

Notation mathématique

Implémentation

Nouvelle formulation

# Pile d'appels

### Visualisation

#### Exponentiation Notion de récursivité

Étude de la

Fonctions Python "built-in"

Préconditions

iviettre en piace

Durée d'exécution

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

récursives

Notation mathématique

### Implémentation

# À retenir

La **pile d'appels** stocke les appels successifs de la fonction récursive.

### Exponentiation Notion de récursivité

Etude de la fonction native

Fonctions Python "built-in"

Préconditions

Durée d'evécution

Duree d'executio

Implémenter la fonction puissance

mathématique Correction de l'algorithme

Correction de l'algorithm

Formulations récursives

Notation mathématique

Implémentation

Nouvelle formulation

# Remarques

▶ Python limite la pile d'appels à 1000 récursions.

```
1 import sys
```

2 sys.setrecursionlimit(20000)

Code 7 – Augmenter le nombre de récursions

### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in"

Préconditions

Durée d'exécution

Implémenter la fonction puissance

mathématique

Correction de l'algorithme

récursives

Notation mathématique

Implémentation

Nouvelle formulation

# Remarques

▶ Python limite la pile d'appels à 1000 récursions.

- 1 | import sys
- 2 sys.setrecursionlimit(20000)

Code 8 – Augmenter le nombre de récursions

La durée d'exécution ne s'est pas améliorée.

1 fonction récursive 0.16802310943603516

### Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in"

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la fonction *puissance* 

mathématique Correction de l'algorithme

Formulations

écursives

Notation mathématique

Implémentation

Nouvelle formulation mathématique

Exponentiation Notion de récursivité

- 1 Étude de la fonction native
- 2. Implémenter la fonction puissance
- 3. Formulations récursives
- 3.1 Notation mathématique
- 3.2 Implémentation
- 3.3 Nouvelle formulation mathématique

### fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des

Durée d'exécution

Implémenter la

S'appuyer sur la définition mathématique

Correction de l'algorithme

Formulations

récursives

Notation mathématique

mplementation

Nouvelle formulation mathématique

# Nouvelle formulation mathématique

FIGURE 1 – Exponentiation rapide

#### Exponentiation Notion de récursivité

Étude de la

Fonctions Python "built-in" Tester un programme

Préconditions Mettre en place des tests

mplémenter la

mathématique

Correction de l'algorithme

ormulations écursives

Notation mathématique

Nouvelle formulation mathématique

puissance(x, n) =

$$\left\{ \begin{array}{ll} 1 & \text{si } \textit{n} = 0 \\ \textit{puissance}(\textit{x} * \textit{x}, \textit{n}/2) & \text{si } \textit{n} > 0 \text{ et n pair} \\ \textit{x.puissance}(\textit{x} * \textit{x}, (\textit{n} - 1)/2) & \text{si } \textit{n} > 0 \text{ et n impair} \end{array} \right.$$

Activité 5 : Implémenter la fonction puissance\_recursif\_rapide(x: int, n: int)→ int qui traduit la formulation récursive précédente.

### Etude de la fonction native

Tester un programme Préconditions

Durée d'exécution

Implémenter la fonction *puissance* 

Correction de l'algorithme

Formulations

récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

```
1
2
3
4
5
```

```
def puissance_recursif_rapide(x: int, n: int) -> int:
    if n == 0: # cas limite
        return 1
    elif n % 2 == 0: # pair
        return puissance_recursif_rapide(x*x, n//2)
    else: # impair
        return x*puissance_recursif_rapide(x*x, n//2)
```

Code 9 - Exponentiation rapide

Visualisation

# Étude de la

Tester un programme
Préconditions

Mettre en place des

Durée d'exécution

# Implémenter la fonction *puissance*

mathématique Correction de l'algorithme

Correction de l'algorithme

#### ormulations écursives

Notation mathématique Implémentation

Nouvelle formulation mathématique

Correction de l'algorithme

Nouvelle formulation

mathématique

1 fonction récursive rapide 0.021007537841796875

Code 10 – Les résultats s'améliorent sans égaler la fonction native.