

Exponentiation
Notion de récursivité

Christophe Viroulaud

Terminale - NSI

Lang 05

Exponentiation Notion de récursivité

Christophe Viroulaud

Terminale - NSI

Lang 05

Exponentiation
Notion de
récursivité

Étude de la
fonction native

Fonctions Python “built-in”

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*

S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

L'exponentiation est une opération mathématique définie
par :

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

Un calcul comme 3^4 ne pose pas de problème mais 2701^{103056} peut prendre un certain à effectuer par le langage de programmation.

L'*exponentiation* est une opération mathématique définie par :

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

► $2^4 \rightarrow 3$ opérations,
 ► $2701^{103056} \rightarrow 103055$ opérations.

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

- $2^4 \rightarrow 3$ opérations,
- $2701^{103056} \rightarrow 103055$ opérations.

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

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

Sommaire

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"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

└ Étude de la fonction native

└ Fonctions Python "built-in"

└ Fonctions Python "built-in"

Fonctions Python "built-in"

```

1 def puissance_star(x:int,n:int)->int:
2     return x**n
3
4 def puissance_builtin(x:int,n:int)->int:
5     return pow(x,n)

```

Code 1 – Fonctions natives

Activité 1 : Tester les deux fonctions du code 1.

Fonctions Python "built-in"

```

1 def puissance_star(x:int,n:int)->int:
2     return x**n
3
4 def puissance_builtin(x:int,n:int)->int:
5     return pow(x,n)

```

Code 1 – Fonctions natives

Activité 1 : Tester les deux fonctions du code 1.

Exponentiation
Notion de
récursivitéÉtude de la
fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Sommaire

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*

3. Formulations récursives

Exponentiation Notion de récursivité

└ Étude de la fonction native

└ Tester un programme

└ Préconditions

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.

Préconditions

Nous décidons de nous limiter au cas positif.

À retenirLa 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
récursivitéÉtude de la
fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

└ Étude de la fonction native

└ Tester un programme

└ Correction

Correction

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

Code 2

Correction

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

Code 2

Exponentiation
Notion de
récursivitéÉtude de la
fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

└ Étude de la fonction native

└ Tester un programme

└ Mettre en place des tests

Mettre en place des tests

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

Mettre en place des tests

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

Exponentiation
Notion de
récursivitéÉtude de la
fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

Étude de la fonction native

Tester un programme

```

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

```

Code 3 – Tester une fonction

```

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

```

Code 3 – Tester une fonction

Exponentiation Notion de récursivité

└ Étude de la fonction native

└ Tester un programme

└ Durée d'exécution

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} .

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
récursivitéÉtude de la
fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

└ Étude de la fonction native

└ Tester un programme

└ Correction

Correction

```
1 from time import time
2
3 debut=time()
4 puissance_star(2701,19406)
5 fin=time()
6 print("opérande **",fin-debut)
```

Correction

```
1 from time import time
2
3 debut=time()
4 puissance_star(2701,19406)
5 fin=time()
6 print("opérande **",fin-debut)
```

Exponentiation
Notion de
récursivitéÉtude de la
fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

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

Sommaire

1. Étude de la fonction native
2. Implémenter la fonction *puissance*
 - 2.1 S'appuyer 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 tests
Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition
mathématique
Correction de l'algorithme

Formulations récursives

Notation mathématique
Implémentation
Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

- Implémenter la fonction *puissance*

- S'appuyer sur la définition mathématique

- S'appuyer sur la définition mathématique

S'appuyer sur la définition mathématique

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

Activité 4 :

1. Implémenter la fonction `puissance_perso(x : int, n : int) → int` sans utiliser les fonctions builtin 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).

S'appuyer sur la définition mathématique

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

Activité 4 :

1. Implémenter la fonction **`puissance_perso(x : int, n : int) → int`** sans utiliser les fonctions builtin 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

— Implémenter la fonction *puissance*

— S'appuyer sur la définition mathématique

— Correction

Correction

```

1 def puissance_perso(x:int,n:int)->int:
2     """
3     >>> puissance_perso(2,8)
4     256
5     >>> puissance_perso(2,9)
6     512
7     """
8     res = 1
9     for i in range(n):
10         res*=x
11     return res

```

```

1 opérande ** 0.006058692932128906
2 fonction pow() 0.005688667297363281
3 fonction personnelle 0.13074541091918945

```

Code 4 – Les résultats sont significatifs.

Correction

```

1 def puissance_perso(x:int,n:int)->int:
2     """
3     >>> puissance_perso(2,8)
4     256
5     >>> puissance_perso(2,9)
6     512
7     """
8     res = 1
9     for i in range(n):
10         res*=x
11     return res

```

```

1 opérande ** 0.006058692932128906
2 fonction pow() 0.005688667297363281
3 fonction personnelle 0.13074541091918945

```

Code 4 – Les résultats sont significatifs.

Sommaire

1. Étude de la fonction native

2. Implémenter la fonction *puissance*

2.1 S'appuyer 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 tests

Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

- Exponentiation Notion de récursivité
 - Implémenter la fonction *puissance*
 - Correction de l'algorithme
 - Correction de l'algorithme

À retenir

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

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

- Implémenter la fonction *puissance*
- Correction de l'algorithme

```

1 res = 1
2 for i in range(n):
3     res*=x

```

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

C'est en fait un raisonnement par récurrence comme en mathématiques.

```

1 res = 1
2 for i in range(n):
3     res*=x

```

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

Étude de la fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

- Implémenter la fonction *puissance*
- Correction de l'algorithme

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

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

Étude de la fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

- Implémenter la fonction *puissance*
- Correction de l'algorithme

► Si $i = 0$ (début de la première itération) la propriété est vérifiée : $x^0 = 1 = res$.
 ► Supposons la propriété vraie au rang p .

- Si $i = 0$ (début de la première itération) la propriété est vérifiée : $x^0 = 1 = res$.
- Supposons la propriété vraie au rang p .

Étude de la fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique

Implémentation

Nouvelle formulation mathématique

Exponentiation Notion de récursivité

- Implémenter la fonction *puissance*
- Correction de l'algorithme

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

► Supposons la propriété vraie au rang p .

► Vérifions au rang $p + 1$:

- au début de l'itération p , $res = x^p$
- à la fin de l'itération p , $res = x^p \times x = x^{p+1}$
- donc au début de l'itération $p+1$, $res = x^{p+1}$

- Si $i = 0$ (début de la première itération) la propriété est vérifiée : $x^0 = 1 = res$.
- Supposons la propriété vraie au rang p .
- Vérifions au rang $p + 1$:
 - au début de l'itération p , $res = x^p$
 - à la fin de l'itération p , $res = x^p \times x = x^{p+1}$
 - donc au début de l'itération $p+1$, $res = x^{p+1}$

Étude de la fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique

Implémentation

Nouvelle formulation mathématique

- 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

Sommaire

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

Étude de la fonction native

Fonctions Python “built-in”
Tester un programme
Préconditions
Mettre en place des tests
Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition
mathématique
Correction de l'algorithme

Formulations récursives

Notation mathématique
Implémentation
Nouvelle formulation
mathématique

Exponentiation Notion de récursivité

- Formulations récursives
- Notation mathématique
- Notation mathématique

récursivité = technique de programmation // impératif

Notation mathématique

$$puissance(x, n) = \begin{cases} 1 & \text{si } n = 0 \\ x.puissance(x, n - 1) & \text{si } n > 0 \end{cases}$$

À retenir

Une fonction **récursive** est une fonction qui s'appelle elle-même.

Notation mathématique

$$puissance(x, n) = \begin{cases} 1 & \text{si } n = 0 \\ x.puissance(x, n - 1) & \text{si } n > 0 \end{cases}$$

À retenir

Une fonction **récursive** est une fonction qui s'appelle elle-même.

Exponentiation Notion de récursivité

Étude de la fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition mathématique

Correction de l'algorithme

Formulations récursives

Notation mathématique

Implémentation

Nouvelle formulation mathématique

Sommaire

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

Étude de la fonction native

Fonctions Python “built-in”
Tester un programme
Préconditions
Mettre en place des tests
Durée d'exécution

Implémenter la fonction *puissance*

S'appuyer sur la définition
mathématique
Correction de l'algorithme

Formulations récursives

Notation mathématique
Implémentation
Nouvelle formulation
mathématique

À retenir

Une fonction récursive :

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

Implémentation

À retenir

Une fonction récursive :

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

```

1 def puissance_recuratif(x: int, n: int) -> int:
2     if n == 0: # cas limite
3         return 1
4     else: # appel récursif
5         return x*puissance_recuratif(x, n-1)

```

Code 6 – Traduction de la formule mathématique

```

1 def puissance_recuratif(x: int, n: int) -> int:
2     if n == 0: # cas limite
3         return 1
4     else: # appel récursif
5         return x*puissance_recuratif(x, n-1)

```

Code 6 – Traduction de la formule mathématique

Pile d'appels

```
main() {
  int a = 1;
  int b = 2;
  int c = 3;
  int d = 4;
  int e = 5;
  int f = 6;
  int g = 7;
  int h = 8;
  int i = 9;
  int j = 10;
  int k = 11;
  int l = 12;
  int m = 13;
  int n = 14;
  int o = 15;
  int p = 16;
  int q = 17;
  int r = 18;
  int s = 19;
  int t = 20;
  int u = 21;
  int v = 22;
  int w = 23;
  int x = 24;
  int y = 25;
  int z = 26;
  int aa = 27;
  int ab = 28;
  int ac = 29;
  int ad = 30;
  int ae = 31;
  int af = 32;
  int ag = 33;
  int ah = 34;
  int ai = 35;
  int aj = 36;
  int ak = 37;
  int al = 38;
  int am = 39;
  int an = 40;
  int ao = 41;
  int ap = 42;
  int aq = 43;
  int ar = 44;
  int as = 45;
  int at = 46;
  int au = 47;
  int av = 48;
  int aw = 49;
  int ax = 50;
  int ay = 51;
  int az = 52;
  int ba = 53;
  int bb = 54;
  int bc = 55;
  int bd = 56;
  int be = 57;
  int bf = 58;
  int bg = 59;
  int bh = 60;
  int bi = 61;
  int bj = 62;
  int bk = 63;
  int bl = 64;
  int bm = 65;
  int bn = 66;
  int bo = 67;
  int bp = 68;
  int bq = 69;
  int br = 70;
  int bs = 71;
  int bt = 72;
  int bu = 73;
  int bv = 74;
  int bw = 75;
  int bx = 76;
  int by = 77;
  int bz = 78;
  int ca = 79;
  int cb = 80;
  int cc = 81;
  int cd = 82;
  int ce = 83;
  int cf = 84;
  int cg = 85;
  int ch = 86;
  int ci = 87;
  int cj = 88;
  int ck = 89;
  int cl = 90;
  int cm = 91;
  int cn = 92;
  int co = 93;
  int cp = 94;
  int cq = 95;
  int cr = 96;
  int cs = 97;
  int ct = 98;
  int cu = 99;
  int cv = 100;
  int cw = 101;
  int cx = 102;
  int cy = 103;
  int cz = 104;
  int da = 105;
  int db = 106;
  int dc = 107;
  int dd = 108;
  int de = 109;
  int df = 110;
  int dg = 111;
  int dh = 112;
  int di = 113;
  int dj = 114;
  int dk = 115;
  int dl = 116;
  int dm = 117;
  int dn = 118;
  int do = 119;
  int dp = 120;
  int dq = 121;
  int dr = 122;
  int ds = 123;
  int dt = 124;
  int du = 125;
  int dv = 126;
  int dw = 127;
  int dx = 128;
  int dy = 129;
  int dz = 130;
  int ea = 131;
  int eb = 132;
  int ec = 133;
  int ed = 134;
  int ee = 135;
  int ef = 136;
  int eg = 137;
  int eh = 138;
  int ei = 139;
  int ej = 140;
  int ek = 141;
  int el = 142;
  int em = 143;
  int en = 144;
  int eo = 145;
  int ep = 146;
  int eq = 147;
  int er = 148;
  int es = 149;
  int et = 150;
  int eu = 151;
  int ev = 152;
  int ew = 153;
  int ex = 154;
  int ey = 155;
  int ez = 156;
  int fa = 157;
  int fb = 158;
  int fc = 159;
  int fd = 160;
  int fe = 161;
  int ff = 162;
  int fg = 163;
  int fh = 164;
  int fi = 165;
  int fj = 166;
  int fk = 167;
  int fl = 168;
  int fm = 169;
  int fn = 170;
  int fo = 171;
  int fp = 172;
  int fq = 173;
  int fr = 174;
  int fs = 175;
  int ft = 176;
  int fu = 177;
  int fv = 178;
  int fw = 179;
  int fx = 180;
  int fy = 181;
  int fz = 182;
  int ga = 183;
  int gb = 184;
  int gc = 185;
  int gd = 186;
  int ge = 187;
  int gf = 188;
  int gg = 189;
  int gh = 190;
  int gi = 191;
  int gj = 192;
  int gk = 193;
  int gl = 194;
  int gm = 195;
  int gn = 196;
  int go = 197;
  int gp = 198;
  int gq = 199;
  int gr = 200;
  int gs = 201;
  int gt = 202;
  int gu = 203;
  int gv = 204;
  int gw = 205;
  int gx = 206;
  int gy = 207;
  int gz = 208;
  int ha = 209;
  int hb = 210;
  int hc = 211;
  int hd = 212;
  int he = 213;
  int hf = 214;
  int hg = 215;
  int hh = 216;
  int hi = 217;
  int hj = 218;
  int hk = 219;
  int hl = 220;
  int hm = 221;
  int hn = 222;
  int ho = 223;
  int hp = 224;
  int hq = 225;
  int hr = 226;
  int hs = 227;
  int ht = 228;
  int hu = 229;
  int hv = 230;
  int hw = 231;
  int hx = 232;
  int hy = 233;
  int hz = 234;
  int ia = 235;
  int ib = 236;
  int ic = 237;
  int id = 238;
  int ie = 239;
  int if = 240;
  int ig = 241;
  int ih = 242;
  int ii = 243;
  int ij = 244;
  int ik = 245;
  int il = 246;
  int im = 247;
  int in = 248;
  int io = 249;
  int ip = 250;
  int iq = 251;
  int ir = 252;
  int is = 253;
  int it = 254;
  int iu = 255;
  int iv = 256;
  int iw = 257;
  int ix = 258;
  int iy = 259;
  int iz = 260;
  int ja = 261;
  int jb = 262;
  int jc = 263;
  int jd = 264;
  int je = 265;
  int jf = 266;
  int jg = 267;
  int jh = 268;
  int ji = 269;
  int jj = 270;
  int jk = 271;
  int jl = 272;
  int jm = 273;
  int jn = 274;
  int jo = 275;
  int jp = 276;
  int jq = 277;
  int jr = 278;
  int js = 279;
  int jt = 280;
  int ju = 281;
  int jv = 282;
  int jw = 283;
  int jx = 284;
  int jy = 285;
  int jz = 286;
  int ka = 287;
  int kb = 288;
  int kc = 289;
  int kd = 290;
  int ke = 291;
  int kf = 292;
  int kg = 293;
  int kh = 294;
  int ki = 295;
  int kj = 296;
  int kk = 297;
  int kl = 298;
  int km = 299;
  int kn = 300;
  int ko = 301;
  int kp = 302;
  int kq = 303;
  int kr = 304;
  int ks = 305;
  int kt = 306;
  int ku = 307;
  int kv = 308;
  int kw = 309;
  int kx = 310;
  int ky = 311;
  int kz = 312;
  int la = 313;
  int lb = 314;
  int lc = 315;
  int ld = 316;
  int le = 317;
  int lf = 318;
  int lg = 319;
  int lh = 320;
  int li = 321;
  int lj = 322;
  int lk = 323;
  int ll = 324;
  int lm = 325;
  int ln = 326;
  int lo = 327;
  int lp = 328;
  int lq = 329;
  int lr = 330;
  int ls = 331;
  int lt = 332;
  int lu = 333;
  int lv = 334;
  int lw = 335;
  int lx = 336;
  int ly = 337;
  int lz = 338;
  int ma = 339;
  int mb = 340;
  int mc = 341;
  int md = 342;
  int me = 343;
  int mf = 344;
  int mg = 345;
  int mh = 346;
  int mi = 347;
  int mj = 348;
  int mk = 349;
  int ml = 350;
  int mn = 351;
  int mo = 352;
  int mp = 353;
  int mq = 354;
  int mr = 355;
  int ms = 356;
  int mt = 357;
  int mu = 358;
  int mv = 359;
  int mw = 360;
  int mx = 361;
  int my = 362;
  int mz = 363;
  int na = 364;
  int nb = 365;
  int nc = 366;
  int nd = 367;
  int ne = 368;
  int nf = 369;
  int ng = 370;
  int nh = 371;
  int ni = 372;
  int nj = 373;
  int nk = 374;
  int nl = 375;
  int nm = 376;
  int nn = 377;
  int no = 378;
  int np = 379;
  int nq = 380;
  int nr = 381;
  int ns = 382;
  int nt = 383;
  int nu = 384;
  int nv = 385;
  int nw = 386;
  int nx = 387;
  int ny = 388;
  int nz = 389;
  int oa = 390;
  int ob = 391;
  int oc = 392;
  int od = 393;
  int oe = 394;
  int of = 395;
  int og = 396;
  int oh = 397;
  int oi = 398;
  int oj = 399;
  int ok = 400;
  int ol = 401;
  int om = 402;
  int on = 403;
  int oo = 404;
  int op = 405;
  int oq = 406;
  int or = 407;
  int os = 408;
  int ot = 409;
  int ou = 410;
  int ov = 411;
  int ow = 412;
  int ox = 413;
  int oy = 414;
  int oz = 415;
  int pa = 416;
  int pb = 41
```

```

puissance_recuratif(6,4)=
    return 6 * puissance_recuratif(6,3)
        |
        return 6 * puissance_recuratif(6,2)
            |
            return 6 * puissance_recuratif(6,1)
                |
                return 6 * puissance_recuratif(6,0)
                    |
                    return 1

```

Visualisation

À retenir

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

À retenir

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

1. Il n'y a pas de raison que ça soit mieux : le nombre d'opérations reste le même
2. même un peu moins bien : la récursivité est moins bien gérée par l'interpréteur Python que par d'autres langages (Ocaml)



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

- Formulations récursives
- Implémentation

1. Il n'y a pas de raison que ça soit mieux : le nombre d'opérations reste le même
2. même un peu moins bien : la récursivité est moins bien gérée par l'interpréteur Python que par d'autres langages (Ocaml)



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
```

- 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

Sommaire

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

Exponentiation
Notion de
récursivité

Durée d'exécution

Correction de l'algorithme

Nouvelle formulation
mathématique

$$a^{2048} = \left(\left(\left(\left(\left(\left(\left(\left(\left(\left((a^2)^2 \right)^2 \right)^2 \right)^2 \right)^2 \right)^2 \right)^2 \right)^2 \right)^2 \right)^2.$$

FIGURE 1 – Exponentiation rapide

$$\text{puissance}(x, n) = \begin{cases} 1 & \text{si } n = 0 \\ \text{puissance}(x * x, n/2) & \text{si } n > 0 \text{ et } n \text{ pair} \\ x * \text{puissance}(x * x, (n-1)/2) & \text{si } n > 0 \text{ et } n \text{ impair} \end{cases}$$

Activité 5 : Implémenter la fonction `puissance_recuris_rapide(x: int, n: int) → int` qui traduit la formulation récursive précédente.

$$\text{puissance}(x, n) =$$

$$\begin{cases} 1 & \text{si } n = 0 \\ \text{puissance}(x * x, n/2) & \text{si } n > 0 \text{ et } n \text{ pair} \\ x * \text{puissance}(x * x, (n-1)/2) & \text{si } n > 0 \text{ et } n \text{ impair} \end{cases}$$

Activité 5 : Implémenter la fonction

`puissance_recuris_rapide(x: int, n: int) → int` qui traduit la formulation récursive précédente.

Exponentiation Notion de récursivité

Formulations récursives

Nouvelle formulation mathématique

Correction

Correction

```

1 def puissance_recuratif_rapide(x: int, n: int) -> int:
2     if n == 0: # cas limite
3         return 1
4     elif n % 2 == 0: # pair
5         return puissance_recuratif_rapide(x*x, n//2)
6     else: # impair
7         return x*puissance_recuratif_rapide(x*x, n//2)

```

Code 9 – Exponentiation rapide

[Visualisation](#)

Correction

```

1 def puissance_recuratif_rapide(x: int, n: int) -> int:
2     if n == 0: # cas limite
3         return 1
4     elif n % 2 == 0: # pair
5         return puissance_recuratif_rapide(x*x, n//2)
6     else: # impair
7         return x*puissance_recuratif_rapide(x*x, n//2)

```

Code 9 – Exponentiation rapide

[Visualisation](#)Exponentiation
Notion de
récursivitéÉtude de la
fonction native

Fonctions Python "built-in"

Tester un programme

Préconditions

Mettre en place des tests

Durée d'exécution

Implémenter la
fonction *puissance*S'appuyer sur la définition
mathématique

Correction de l'algorithme

Formulations
récursives

Notation mathématique

Implémentation

Nouvelle formulation
mathématique

1. Implémentation des fonctions builtin en C
2. itératif plus rapide car appels fonction coûtent ; mais récursif donne souvent code plus clair/lisible

```
1 fonction récursive rapide
  0.021007537841796875
```

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

```
1 fonction récursive rapide
  0.021007537841796875
```

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