

Fonction produit [rc02] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel  algoprogram  Version 21 mai 2018

Table des matières

1	Fonction produit / pgproduit	2
1.1	Produit naïf	2
1.2	Stratégie basée sur la parité	2
1.3	Autre stratégie basée sur la parité	3
1.4	Programme de test	4
2	Références générales	4

Python - Fonction produit (Solution)



Mots-Clés Récursivité des actions ■

Requis Algorithmes paramétrés ■

Difficulté ● ○ ○



Objectif

Cet exercice calcule récursivement le produit $a \cdot n$ d'un réel a par un entier $n \geq 0$ de plusieurs manières. Dans le même ordre d'idées, l'exercice @[Fonction puissance] calcule récursivement la puissance x^n d'un réel x par un entier $n \geq 0$ de plusieurs manières.

1 Fonction produit / pgproduit

1.1 Produit naïf

Soient un entier $n \geq 0$ et un réel a .

L'idée la plus simple pour calculer le produit de a par n consiste à se dire :

- si $n = 0$ alors :

$$P(a, n) = 0$$

- si $n \geq 1$ et si je sais calculer $P(a, n - 1)$ alors :

$$\begin{aligned} P(a, n) &= a \cdot n \\ &= a \cdot (n - 1) + a \\ &= P(a, n - 1) + a \end{aligned}$$



Écrivez une fonction récursive `produit1(a,n)` qui traduit cette analyse.



Validez votre fonction avec la solution.

Solution Python @[pgproduit.py]

```
def produit1(a, n):
    """ Produit naïf (suppose n >= 0)

    :param a: un réel
    :param n: un entier
    :return: Produit a*n
    """
    return (0.0 if n == 0 else produit1(a, n - 1) + a)
```

1.2 Stratégie basée sur la parité

L'idée basée sur la méthode « diviser pour régner » consiste à se dire : supposons que l'on sache calculer le produit de a par $n \text{ div } 2$. Comment calculer alors le produit de a par n ?

Cela donne naissance à l'analyse suivante :

- si $n = 0$ alors :

$$P(a, n) = 0$$

- si n est pair et $n \geq 2$, alors si je sais calculer $P(a, n \text{ div } 2)$, on a :

$$P(a, n) = P(a, n \text{ div } 2) + P(a, n \text{ div } 2)$$

- si n est impair et $n \geq 1$, alors $n - 1$ est pair et donc :

$$P(a, n) = P(a, (n - 1) \text{ div } 2) + P(a, (n - 1) \text{ div } 2) + a$$



Écrivez une fonction récursive `produit2(a,n)` qui traduit cette analyse.



Validez votre fonction avec la solution.

Solution Python @[pgproduit.py]

```
def produit2(a, n):
    """ Produit basé sur la parité (suppose n >= 0)

    :param a: un réel
    :param n: un entier
    :return: Produit a*n
    """
    if n == 0:
        return 0.0
    else:
        p = produit2(a, n // 2)
        return p + p + (0 if n % 2 == 0 else a)
```

1.3 Autre stratégie basée sur la parité

Une idée similaire à celle de la deuxième méthode mais basée sur l'hypothèse que l'on sait calculer $P(2a, n \text{ div } 2)$ est la suivante :

- si n est nul alors :

$$P(a, n) = 0$$

- si n est pair et $n \geq 2$, alors :

$$P(a, n) = P(2a, n \text{ div } 2)$$

- si n est impair et $n \geq 1$, alors $n - 1$ est impair, d'où :

$$P(a, n) = P(2a, (n - 1) \text{ div } 2) + a$$



Écrivez une fonction récursive `produit3(a,n)` qui traduit cette analyse.



Validez votre fonction avec la solution.

Solution Python @[pgproduit.py]

```
def produit3(a, n):
    """ Produit basé sur la parité (suppose n >= 0)

    :param a: un réel
    :param n: un entier
    :return: Produit a*n
    """
    if n == 0:
```

```

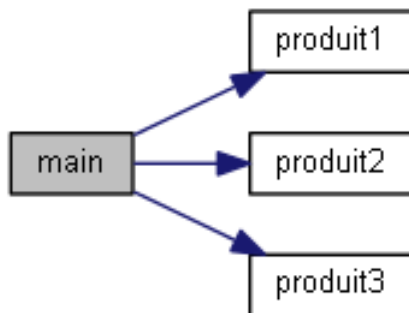
    return 0.0
else:
    p = produit3(a + a, n // 2)
    return p + (0 if n % 2 == 0 else a)

```

1.4 Programme de test



Écrivez un script qui saisit un réel dans `x` et un entier dans `n` puis calcule et affiche le résultat des fonctions `produit1(x,n)`, `produit2(x,n)` et `produit3(x,n)`.



Testez. Exemple d'exécution :

```

Un réel x? 1.5
Un entier n? 97
==> produit1(x,n) vaut 145.5
==> produit2(x,n) vaut 145.5
==> produit3(x,n) vaut 145.5

```



Validez votre script avec la solution.

Solution Python @[pgproduit.py]

```

def PGProduit():
    x = float(input("Un reel x? "))
    n = int(input("Un entier n? "))
    print("==> produit1(x,n) vaut ", produit1(x, n), sep="")
    print("==> produit2(x,n) vaut ", produit2(x, n), sep="")
    print("==> produit3(x,n) vaut ", produit3(x, n), sep="")

```

2 Références générales

Comprend ■