

Divisibilité et nombres premiers

Compétences visées

- **chercher**, expérimenter – en particulier à l'aide d'outils logiciels ;
- **modéliser**, faire une simulation, valider ou invalider un modèle ;
- **représenter**, choisir un cadre (numérique, algébrique, géométrique...), changer de registre ;
- **calculer**, appliquer des techniques et mettre en œuvre des algorithmes.

Ces compétences sont mises en œuvre dans le cadre de l'extrait du programme de 2^{nde} GT ci-dessous :

- « Déterminer si un entier naturel a non nul est multiple d'un entier naturel b non nul. »
- « Déterminer si un entier naturel est premier. »

Situation déclenchante



L'os d'Ishango

Les entailles retrouvées sur l'os d'Ishango daté à plus de 20 000 ans avant notre ère, semblent isoler quatre nombres premiers 11, 13, 17 et 19. Certains archéologues l'interprètent comme la preuve de la connaissance des nombres premiers. Toutefois, il existe trop peu de découvertes permettant de cerner les connaissances réelles de cette période ancienne.

Problématique

Ecrire un script qui permet de vérifier si un nombre est premier ou non.

On pourra ensuite écrire un script qui fournit la décomposition en produit de facteurs premiers d'un nombre entier.

Fiche méthode

Proposition de résolution

On crée trois fonctions dans ce script :

- Une fonction **diviseur** qui prend comme arguments deux entiers naturels a et b et qui renvoie vrai si b divise a et faux sinon.
- Une fonction **liste** qui prend comme argument un entier naturel a et qui renvoie la liste des diviseurs positifs de a.
- Une fonction **premier** qui prend comme argument un entier naturel p et qui renvoie vrai si p est un nombre premier et faux sinon.

```
PYTHON SHELL
>>> diviseur(15,5)
True
>>> diviseur(15,6)
False
>>> diviseur(18,9)
True
>>> |
Fns... a A # Outils Éditer Script
```

```
PYTHON SHELL
>>> liste(12)
[1, 2, 3, 4, 6, 12]
>>> liste(17)
[1, 17]
>>> liste(18)
[1, 2, 3, 6, 9, 18]
>>> |
Fns... a A # Outils Éditer Script
```

```
PYTHON SHELL
>>> premier(5)
True
>>> premier(15)
False
>>> premier(1)
False
>>> premier(2)
True
>>> premier(12)
False
>>> |
Fns... a A # Outils Éditer Script
```

Etapes de résolution

La fonction **diviseur** prend comme arguments deux entiers naturels a et b et renvoie vrai ou faux si b divise a ou non. L'instruction `a%b` permet de déterminer le reste dans la division euclidienne de a par b.

```
ÉDITEUR : PREMIER
LIGNE DU SCRIPT 0007
def diviseur(a,b):
    if a%b==0:
        return True
    else:
        return False
-
Fns... a A # Outils Exéc Script
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus



Fiche méthode

Etapes de résolution

C'est un principe à retenir : On peut appeler une fonction (ici : la fonction **diviseur**) à l'intérieur d'une autre fonction (ici : **liste**). L'utilisation successive de fonctions en python rend le script dans son ensemble plus lisible.

La fonction **liste** prend comme argument un entier naturel a et renvoie la liste des diviseurs de a . L'instruction `l=[]` permet d'initialiser la liste `l` avec une liste vide. L'instruction `l.append(i)` permet de rajouter le nombre i à la liste `l`.

La fonction **premier** prend comme argument un entier naturel p et renvoie vrai si p est un nombre premier et faux sinon. On fait appel à la fonction **liste** pour vérifier si p admet exactement deux diviseurs 1 et p .

```
EDITEUR : PREMIER
LIGNE DU SCRIPT 0016

def liste(a):
    l=[]
    for i in range(1,a+1):
        if diviseur(a,i)==True:
            l.append(i)
    return l

Fns... a A # Outils Exéc Script
```

```
EDITEUR : PREMIER
LIGNE DU SCRIPT 0023

def premier(p):
    if liste(p)==[1,p]:
        return True
    else:
        return False

Fns... a A # Outils Exéc Script
```

```
PYTHON SHELL

>>> premier(5)
True
>>> premier(15)
False
>>> premier(1)
False
>>> premier(2)
True
>>> premier(12)
False
>>> |

Fns... a A # Outils Éditer Script
```

```
EDITEUR : PREMIER
LIGNE DU SCRIPT 0046

def premier2(p):
    if p==1:
        return False
    for i in range(2,p):
        if p%i==0:
            return False
    return True

Fns... a A # Outils Exéc Script
```

Deuxième méthode

Sans utiliser les listes, on peut créer d'une autre manière une fonction **premier2** qui prend comme argument un entier naturel p et qui renvoie vrai si p est premier et faux sinon.

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus



Fiche méthode

Approfondissement possible

En approfondissement, on peut travailler sur la décomposition en produit de facteurs premiers d'un entier naturel. La fonction **decomposition** qui prend comme argument un entier naturel n et qui renvoie la liste des nombres premiers dans la décomposition en facteur premier de n , ainsi que la liste des puissances associées dans cette décomposition de n .

Cette boucle permet de créer la liste des diviseurs premiers de n .

Initialisation de la liste des puissances à 1.
L'instruction `len(p)` permet de renvoyer la longueur de la liste p .

Cette boucle permet de tester la divisibilité de n par les puissances des diviseurs premiers de n et de stocker cette puissance dans la liste `puissance`.

```
ÉDITEUR : PREMIER
LIGNE DU SCRIPT 0031
def decomposition(n):
    l=liste(n)
    p=[]
    for i in l:
        if premier(i)==True:
            p.append(i)
            puissance=[1]*(len(p))
            for j in range(0,len(p)):
                k=2
                while n%(p[j]**k)==0:
                    puissance[j]=k
                    k=k+1
    return (p,puissance)

Fns... a A # Outils Exéc Script
```

Remarque : L'instruction `range(0, len(p))` aurait pu être remplacée par `range(len(p))` car par défaut l'instruction `range` commence à 0.

```
PYTHON SHELL
>>> decomposition(126)
[[2, 3, 7], [1, 2, 1]]
>>> |

Fns... a A # Outils Éditer Script
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

