

Test de primalité

| | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|-----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

ARE Calcul et raisonnement

2015-2016

Fatemeh HAMISSI

Table des matières :

| | |
|---|----|
| Introduction..... | 3 |
| La division euclidienne et divisibilité | 4 |
| Les nombres premiers..... | 8 |
| Calcul de PGCD..... | 9 |
| Congruence..... | 12 |
| Théorème de Fermat..... | 14 |
| Quelques autres test de primalité..... | 17 |
| Les algorithmes..... | 19 |

Introduction :

Les nombres premiers ont un rôle important en mathématiques grâce à leurs propriétés très utiles dans diverses applications particulièrement dans le domaine de la cryptographie.

Un test de primalité est un algorithme permettant de savoir si un nombre entier est premier.

Au XVII^e siècle Fermat, le mathématicien français a trouvé un théorème nommé petit théorème de Fermat sur les propriétés de ces nombres, la réciproque de ce théorème peut être utilisée comme un test de primalité.

L'étude de ce théorème et de toutes les notions mathématiques nécessaires pour le comprendre et l'utiliser fera l'objet principal de cet ARE.

1-La division euclidienne

Une division euclidienne est une opération qui, à deux entiers naturels (appelés dividende et diviseur), associe deux autres entiers (appelés quotient et reste). Cette division est définie initialement pour deux entiers naturels non nuls, elle se généralise aux entiers relatifs.

Théorème1-1 :

Soit a et b deux entiers naturels, avec b non nul. Il existe un couple unique d'entiers naturels (q, r) tel que :

$$a = bq + r \text{ et } 0 \leq r < b$$

Exemples :

$$31=4 \times 7 + 3$$

$$19=2 \times 9 + 1$$

L'entier a est appelé le **dividende** de cette division, b le **diviseur**, q le **quotient**, r le **reste**.

2-Divisibilité

Définition :

Un entier $a \in \mathbb{Z}$ est dit divisible par un entier $b \in \mathbb{Z}$ (b non nul), s'il existe un entier k tel que :

$$a = bk. \text{ (Donc } r=0 \text{)}$$

On dit alors que a est un multiple de b ou b divise a ou est un diviseur de a . b divise a se note b/a .

Exemples : $45 = 9 \times 5$, $26 = 13 \times 2$

Quelques propriétés :

2-1) 0 est divisible par tout entier b .

2-2) 1 et -1 divisent tout entier relatifs.

2-3) $-a|b$, $a|-b$, $-a|-b$

La relation « divise » est une relation d'ordre partiel ; en effet la relation est

- Réflexive : a/a

- Transitive : si a/b et b/c alors a/c
- Antisymétrique : si a/b et b/a alors $a = b$.

On peut extraire quelques lois générales qui nous dit si un entier est divisible par l'autre ou pas.
On va l'expliquer dans la partie congruence.

Calcul de PGCD

Définition :

Le plus grand commun diviseur, de deux nombres entiers relatifs (où au moins l'un de ces deux est nul.) est le plus grand entier qui divise simultanément ces deux entiers.

Exemple : Les diviseurs communs de 24 et 84 sont : $\pm 1, \pm 2, \pm 3, \pm 4, \pm 6, \pm 12$

Donc $\text{PGCD}(24, 84) = 12$

Théorème 4-1 : l'entier naturel d est le PGCD de deux entiers relatifs a et b si et seulement si :

- 1) $d|a$ et $d|b$
- 2) Quand $c|a$ et $c|b$ alors $c|d$

Exemple : pour $a=24$ et $b=84$ et $c=2$ on a : $2|24, 2|84 \rightarrow 2|\text{PGCD}(24, 84) \rightarrow 2|12$

Les nombres premiers :

Définition :

Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs, 1 et lui-même.

Un nombre composé est un nombre qui n'est pas premier (sauf 1 qui n'est ni premier ni composé par ce qu'il admet un seul diviseur qui est donc 1, 0 non plus car il est divisible par tous les entiers positifs).

Exemple : $10 = 2 \times 5$ est composé, tout comme $48 = 6 \times 8$, mais 13 est premier car 1 et 13 sont les seuls diviseurs de 13.

Théorème 3-1 : Tout entier relatifs a au moins un diviseur premier.

Théorème 3-2 : il y a une infinité de nombres premiers.

Théorème 3-3 : Si n est un nombre composé alors n admet au moins un diviseur premier inférieur ou égale à \sqrt{n} .

Exemples : on pose $n=47$,

$6 < \sqrt{47} < 7$ donc les nombres premiers inférieur à $\sqrt{47}$ sont 2,3,5 et nul d'entre eux divisent 47 donc 47 est un nombre premier.

Théorème 3-4 : Théorème fondamental de l'arithmétique

Tout entier strictement positif peut être écrit comme un produit de nombres premiers d'une unique façon, à l'ordre près des facteurs.

Exemples : $8=2^3$ $72 = 2^3 \times 3^2$ $6\,936 = 2^3 \times 3 \times 17^2$

Il existe plusieurs méthodes pour trouver le PGCD de deux nombres mais l'un de ces méthodes utilise le théorème fondamental de l'arithmétique (théorème 3-4) :

Dans cette méthode on factorise les deux nombres par leur diviseurs premiers (comme les exemples situés en haut) le PGCD est la multiplie des facteurs communs avec leur plus petite exposant.

Exemples : On veut calculer le PGCD de 30 et 72, on a :

$$30 = 2^1 \times 3^1 \times 5^1 \qquad 72 = 2^3 \times 3^2$$

Donc les diviseurs communs sont 2 et 3 et le PGCD est : $2^1 \times 3^1 = 6$

Les nombres premiers entre eux :

Deux nombres entiers relatifs sont dit premiers entre si leur PGCD=1.

Exemples :

9 et 10 sont premiers entre eux puisque les diviseurs de 9 sont : 1, 3, 9 et ceux de 10 sont 1, 2, 5, 10 et donc le PGCD est le 1.

10 et 15 ne sont pas premiers entre eux puisque les diviseurs de 10 sont : 1, 2, 5, 10 et ceux de 15 sont 1, 3, 5, 15 et donc le PGCD est le 5.

Théorème 4-2 : Théorème de Bézout

a et b sont premiers entre eux s'il existe deux entiers relatifs x et y tels que

$$ax + by = 1$$

Exemples : on pose $a=3$ $b=2$ on a $x=5$ et $y=-7$ puisque : $(3)(5) + (2)(-7) = 1$

Lemme de Gauss : Si un nombre entier a divise le produit de deux autres nombres entiers b et c , et si a est premier avec b , alors a divise c .

Formellement :

$$\forall (a, b, c) \in \mathbb{Z}^3, (a|bc \text{ et } PGCD(a, b) = 1) \Rightarrow a|c$$

L'indicatrice d'Euler :

L'indicatrice d'Euler est une fonction, qui à tout entier naturel n non nul associe le nombre d'entiers compris entre 1 et n (inclus) et premiers avec n .

Exemples :

$\varphi(5)$ = les entiers inférieurs et premiers à 5 sont : 1, 2, 3, 4 donc on a 4 entiers ; $\varphi(5) = 4$.

$\varphi(8)$ = les entiers inférieurs et premiers à 8 sont : 1, 3, 5, 7 donc on a 4 entiers ; $\varphi(8) = 4$.

Si on a p un nombre premier alors comme p est premier à tous les nombres inférieurs à lui $\varphi(p) = p - 1$.

On veut maintenant démontrer la réciproque :

On a $\varphi(n) = n - 1$ qui veut dire que tous les entiers inférieurs à n sont premiers à n (car il y en a $n - 1$). Or si n n'est pas premier il y a au moins un entier inférieur à lui qui n'est pas premier à lui. Contradiction

Donc le nombre est premier.

Congruence :

Deux nombres sont congrus par un module quelconque quand ce module divise leurs différence

$$a \equiv b \pmod{m} \text{ quand } m|(a - b)$$

$$a, b \in \mathbb{Z} \quad m \in \mathbb{N}$$

La congruence (modulo n) a les propriétés suivantes :

Réflexivité : pour tout entier a, $a \equiv a \pmod{n}$;

Symétrie : pour tous entiers a et b, $a \equiv b \pmod{n} \Leftrightarrow b \equiv a \pmod{n}$;

Transitivité : pour tous entiers a, b et c, si $a \equiv b \pmod{n}$ et $b \equiv c \pmod{n}$ alors $a \equiv c \pmod{n}$.

Il s'agit donc d'une relation d'équivalence et donc cette relation d'équivalence divise l'ensemble \mathbb{Z} par des classes d'équivalence.

Soit [a] une classe d'équivalence et a un élément de cette classe soit b un autre élément de cette classe d'équivalence on a :

$[a]=[b]$ si et seulement si $a \equiv b \pmod{m}$ par exemple dans congruence (mod 6) on a :

$$[15] = [3] = [-3] = \dots$$

$$[-5] = [1] = [7] = \dots$$

Quelques propriétés de congruence :

1-Si $a \equiv b \pmod{m}$ alors pour tous entier relatifs c :

$$a + c \equiv b + c \pmod{m}$$

2-Si $a \equiv b \pmod{m}$ et $c \equiv d \pmod{m}$ alors :

$$a + c \equiv b + d \pmod{m}$$

$$ac \equiv bd \pmod{m}$$

3-Si on a $a_1 \equiv b_1 \pmod{m}$ et ...et $a_n \equiv b_n \pmod{m}$ alors :

$$a_1 + a_2 + \dots + a_n \equiv b_1 + b_2 + \dots + b_n \pmod{m}$$

$$a_1 a_2 \dots a_n \equiv b_1 b_2 \dots b_n \pmod{m}$$

4-Si $a \equiv b \pmod{m}$ alors pour tout $n \geq 1$ on a :

$$a^n \equiv b^n \pmod{m}$$

Exemple : On veut calculer $2^{15} = x \pmod{100}$:

$$2^7 = 128 \equiv 28 \pmod{100}$$

$$2^{14} = 28^2 = 784 \equiv 84 \pmod{100}$$

$$84 \times 2 = 168 \equiv 68 \pmod{100}$$

$$\text{Donc } x = 68$$

Théorème 5-1 :

Si on a $ac \equiv bc \pmod{m}$ alors :

$$a \equiv b \pmod{\frac{m}{d}}$$

Où d est le PGCD de m et c.

Exemple : $8 \equiv 2 \pmod{6} \rightarrow 4 \equiv 1 \pmod{3}$

Maintenant à l'aide de la définition de la congruence on peut définir quelques lois générales sur la divisibilité des nombres.

| | | |
|-------------------------|--|--|
| Divisible par 2 | Un nombre est divisible par 2 si son dernier chiffre est divisible par 2. | 1 274 est divisible par 2, car 4 est divisible par 2. |
| Divisible par 3 | Un nombre est divisible par 3 si la somme de ses chiffres est divisible par 3. | 294 est divisible par 3, car $2 + 9 + 4 = 15$ et 15 est divisible par 3. |
| Divisible par 9 | Un nombre est divisible par 9 si la somme de ses chiffres est divisible par 9. | 945 est divisible par 9, car $9 + 4 + 5 = 18$ et 18 est divisible par 9. |
| Divisible par 10 | Un nombre est divisible par 10 s'il se termine par 0. | 6 680 est divisible par 10, car il se termine par un 0. |

Démonstration de la divisibilité par 2 :

On sait qu'un nombre $A = (a_n a_{n-1} \dots a_1 a_0)_{10} = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \dots + a_1 \times 10 + a_0$

D'autre part on sait que $10^n \equiv 0 \pmod{2}$ donc $a_n \times 10^n \equiv 0 \pmod{2}$ ainsi tous les $a_i \times 10^i$ sont divisible par 2 et il ne reste que a_0 , donc pour que le nombre soit divisible par 2 il suffit que son dernier chiffre (a_0) soit divisible par 2.

On a la même loi pour la divisibilité par 10 puisque $10^n \equiv 0 \pmod{10}$

Donc le dernier chiffre doit être 0.

Démonstration de la divisibilité par 3 :

On sait que $10^n \equiv 1 \pmod{3}$ donc on a $a_n \times 10^n \equiv 1 \times a_n \pmod{3}$ et donc le reste de la division de nombre A par 3 est :

$$a_n + a_{n-1} + \dots + a_1 + a_0$$

Donc si on veut que le nombre A soit divisible par 3 ($a_n + a_{n-1} + \dots + a_1 + a_0$) doit être divisible par 3. Ce qui veut dire la somme des chiffres doit être divisible par 3.

On a la même loi pour la divisibilité par 9 puisque $a_n \times 10^n \equiv a_n \pmod{9}$ et il suffit que la somme des chiffres soit divisible par 9.

Théorème de Fermat

Définition :

Soit p un nombre premier quelconque et a un nombre entier quelconque on a :

$$a^p \equiv a \pmod{p}$$

La deuxième forme de ce théorème s'écrit ainsi lorsque a et p sont premiers entre eux :

$$a^{p-1} \equiv 1 \pmod{p}$$

$$a \in \mathbb{Z}, m \in \mathbb{N}$$

Démonstration :

On démontre par récurrence :

On sait que pour $a=0$ ou $a=1$ quel que soit p la congruence est vraie.

On suppose que $a^p \equiv a \pmod{p}$ est vrai et donc $a^p - a \equiv 0 \pmod{p}$ est vrai.

On démontre que la proposition a lieu en augmentant a d'une valeur 1 et donc

$$(a+1)^p - (a+1) \equiv 0 \pmod{p}$$

On sait que $(a+1)^p = \sum_{k=0}^p \binom{p}{k} a^k$ comme $\binom{p}{k} = \frac{p!}{k!(p-k)!}$ avec

$k \in [1, p-1]$ $k < p$ et $p-k < p$ En écrivent:

$p \times (p-1)! = \binom{p}{k} \times k! (p-k)!$, On a directement $p | \binom{p}{k} k! (p-k)!$ Puisque p est premier il n'est pas divisible par les facteurs de $k!$ ou $(p-k)!$ c'est à dire que $\text{pgcd}(p, k!(p-k)!) = 1$. donc d'après le théorème de Gauss $\binom{p}{k}$ est divisible par p donc tous les coefficients $(a+1)^p$ sont divisibles par p et donc congrus à 0 (mod p) on peut dire $(a+1)^p \equiv a^p + 1$

(mod p) en suite on soustrait les 2 coté par $a+1$ donc on

a :

$$(a+1)^p - (a+1) \equiv a^p - a \pmod{p}$$

Comme on avait $a^p - a \equiv 0 \pmod{p}$ donc $(a+1)^p - (a+1) \equiv 0 \pmod{p}$.

Démonstration pour la deuxième forme :

En factorisant $a^p - a$ par a on a :

$$a^p - a = a(a^{p-1} - 1)$$

On sait que p divise $a^p - a$ et on sait aussi que p et a sont premiers entre eux donc p divise l'autre facteur c'est-à-dire $(a^{p-1} - 1)$ ainsi on a :

$$a^{p-1} \equiv 1 \pmod{p}$$

Théorème d'Euler (théorème de Fermat généralisé)

Soit n un entier strictement positif et a un entier premier avec n alors :

$$a^{\varphi(n)} \equiv 1 \pmod{n}$$

Où $\varphi(n)$ est l'indicateur d'Euler.

Gaussien d'un nombre :

A partir du théorème d'Euler on sait que qu'il existe de puissance de a qui donne 1 pour reste soit g le plus petit exposant tel que $a^g \equiv 1 \pmod{n}$ on dit alors que a appartient à l'exposant minimum g pour le module n ou on dit que g est le gaussien de a pour le module n .

La réciproque fautive du théorème de Fermat :

La réciproque du théorème de Fermat qui est la suivante :

Pour qu'un entier $p \geq 2$ soit premier, il faut que :

$$\forall a \in \{1, 2, \dots, p-1\}, a^{p-1} \equiv 1 \pmod{p}$$

$$a \in \mathbb{Z} \quad m \in \mathbb{N}$$

Il existe des contre-exemples qui prouvent que cette réciproque est fautive.

Exemples : $561 = 3 \times 11 \times 17$ n'est pas premier et pourtant pour tout entier a premier avec 561, on a $a^{561} \equiv 1 \pmod{561}$.

La réciproque valable du théorème de Fermat :

Pour que la réciproque de ce théorème soit valable il faut ajouter quelques conditions. Donc la bonne réciproque de théorème de Fermat est :

S'il existe a premier à n tel que on a :

$$a^{n-1} \equiv 1 \pmod{n} \quad \text{et} \quad a^{\frac{n-1}{p}} \not\equiv 1 \pmod{n} \quad \forall p|(n-1)$$

Alors n est premier.

Démonstration :

Comme pour tous diviseur de $n-1$ sauf lui-même le nombre n n'est pas congrue à $1 \pmod{n}$ on déduit que $n-1$ est le plus petit g tel que :

$a^g \equiv 1 \pmod{n}$ et donc d'après la définition de gaussien, $n-1$ est le gaussien de $a \pmod{n}$.

D'autre part d'après théorème d'Euler on a $a^{\varphi(n)} \equiv 1 \pmod{n}$ on sait que $n-1$ est le gaussien de $a \pmod{n}$ donc $n-1 < \varphi(n)$ mais comme de la définition de $\varphi(n)$ on a $1 \leq \varphi(n) \leq n-1$, alors : $\varphi(n) \leq n-1$.

Donc comme $n-1$ est à la fois supérieur et inférieur à $\varphi(n)$ donc il est forcément égal à $n-1$.

Et comme on sait que si $\varphi(n)$ est égale à $n-1$ alors n est forcément un nombre premier (voir la partie 'les nombres premier'), alors n est premier.

Exemple :

On prend $n=5$; les nombres premiers à n (les a) sont : 2,3,4 et le nombre qui divise $n-1$ est 2.

Donc on doit vérifier que : $2^2 \not\equiv 1 \pmod{5}$ ce qui est le cas

$$3^2 \not\equiv 1 \pmod{5} \text{ ce qui est le cas}$$

$$4^2 \not\equiv 1 \pmod{5} \text{ ce qui n'est pas le cas}$$

Donc on test la condition suivante pour $a=2$ ou $a=3$

$$\text{Pour } 2 : \quad 2^4 \equiv 1 \pmod{5} \text{ ce qui est le cas}$$

$$\text{Pour } 3 : \quad 3^4 \equiv 1 \pmod{5} \text{ ce qui est le cas}$$

Donc 5 est bien premier.

Quelques autres tests de primalité

Méthode naïve

Le test le plus simple est de vérifier si le nombre N est divisible par l'un des entiers compris entre 2 et N (2 compris). Si la réponse est négative, alors N est premier, sinon il est composé.

Plusieurs changements permettent d'améliorer les performances de cet algorithme :

Il suffit de tester tous les nombres de 2 à \sqrt{N} seulement, puisque si $N = pq$ alors soit $p \leq \sqrt{N}$ soit $q \leq \sqrt{N}$.

Crible d'Ératosthène :

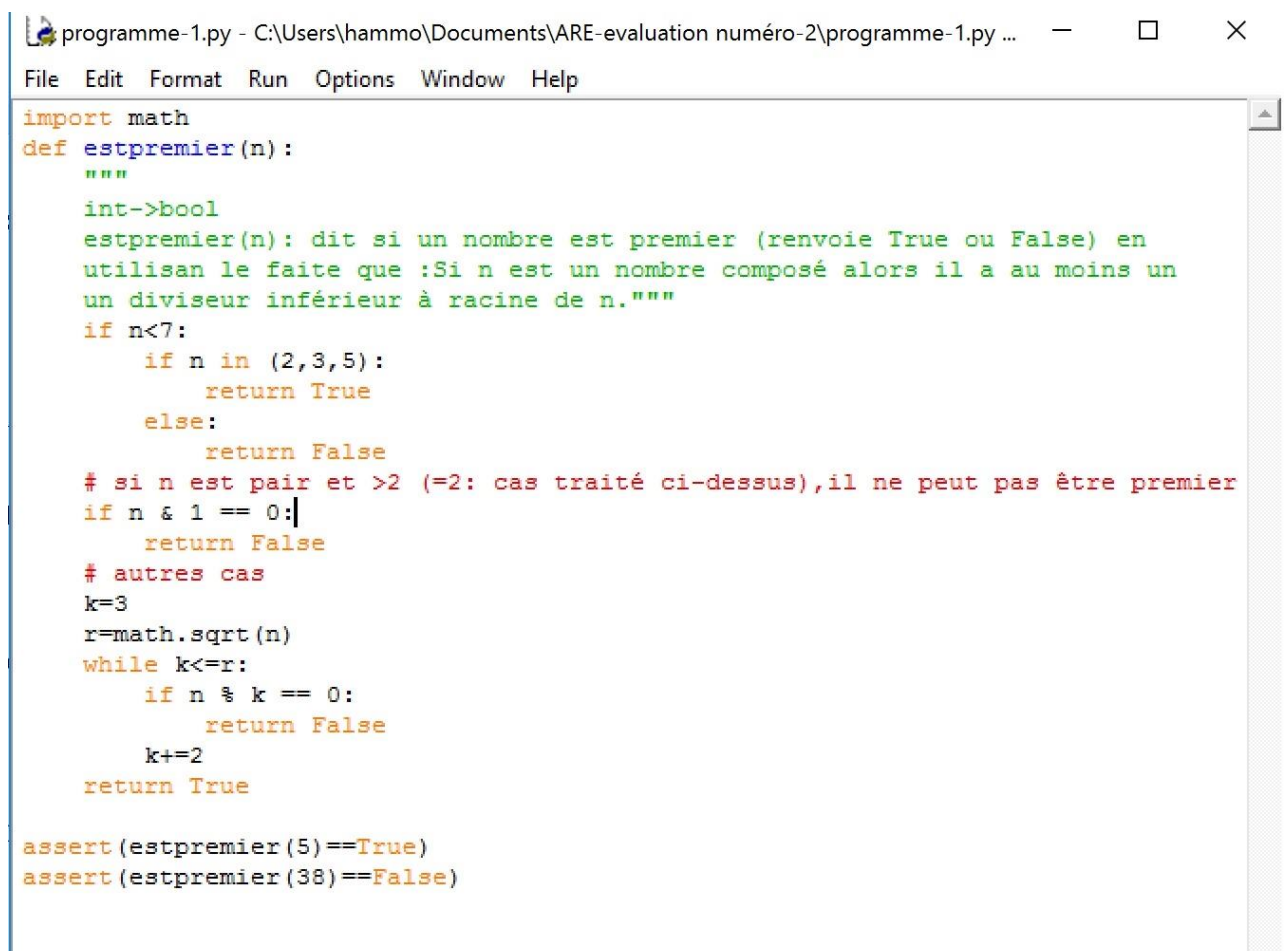
C'est un algorithme qui procède par élimination : il s'agit de supprimer d'une table des entiers de 2 à N tous les multiples d'un entier. Donc à la fin il ne restera que les entiers qui ne sont multiples d'aucun entier, et qui sont donc les nombres premiers.

Les algorithmes

Dans cette partie on va utiliser les notions mathématiques qu'on a vu pour pouvoir écrire des programmes informatiques. Le but principal est de faire un test de primalité qui est basé sur la réciproque de théorème de Fermat.

1-Programme n°1

Un test de primalité naïf qui utilise la propriété décrit dans la partie 'Quelques autres tests de primalité'



```
programme-1.py - C:\Users\hammo\Documents\ARE-evaluation numéro-2\programme-1.py ...
File Edit Format Run Options Window Help

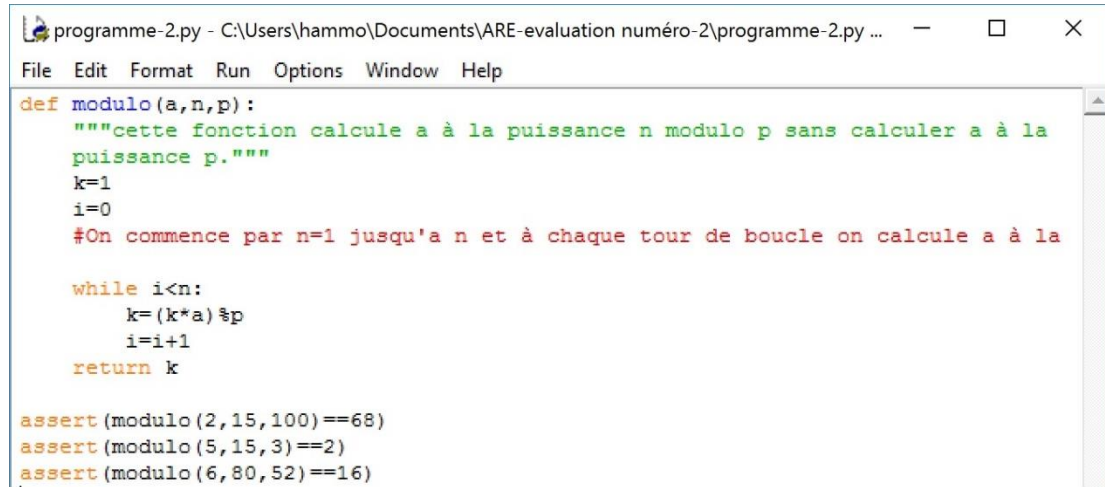
import math
def estpremier(n):
    """
    int->bool
    estpremier(n): dit si un nombre est premier (renvoie True ou False) en
    utilisant le fait que :Si n est un nombre composé alors il a au moins un
    un diviseur inférieur à racine de n."""
    if n<7:
        if n in (2,3,5):
            return True
        else:
            return False
    # si n est pair et >2 (=2: cas traité ci-dessus), il ne peut pas être premier
    if n & 1 == 0:
        return False
    # autres cas
    k=3
    r=math.sqrt(n)
    while k<=r:
        if n % k == 0:
            return False
        k+=2
    return True

assert(estpremier(5)==True)
assert(estpremier(38)==False)
```

2-Programme n°2

Programme qui calcule l'inconnu x dans la congruence $a^n \equiv x \pmod{p}$ sans calculer a^n .

Ne pas calculer a^n nous aide à économiser du temps et de la mémoire.



```
programme-2.py - C:\Users\hammo\Documents\ARE-evaluation numéro-2\programme-2.py ...
File Edit Format Run Options Window Help

def modulo(a,n,p):
    """cette fonction calcule a à la puissance n modulo p sans calculer a à la
    puissance p."""
    k=1
    i=0
    #On commence par n=1 jusqu'a n et à chaque tour de boucle on calcule a à la

    while i<n:
        k=(k*a)%p
        i=i+1
    return k

assert(modulo(2,15,100)==68)
assert(modulo(5,15,3)==2)
assert(modulo(6,80,52)==16)
```

Pour ce programme on va à l'aide d'un invariant démontrer que le programme retourne bien le résultat souhaité. On veut montrer qu'après chaque tour de boucle le programme retourne $a^i \% p = res$

Démonstration par récurrence :

Initialisation :

$$i=0$$

$$res=a^0 \bmod p$$

$$1=a^0 \bmod p \rightarrow 1 \bmod p \text{ (vrai)}$$

- Hérité :

Supposons $P(i) : res = a^i \bmod p$:

Après premier instruction $i=i+1$ on a donc $res=a^{i-1} \pmod{p}$

deuxième instruction:

$$res = (a * res) \% p$$

or

$$res = (a * (a^{i-1} \% p)) \% p$$

On sait que $(a^{i-1} \% p) = x$ en langage informatique est égale à $a^{i-1} \equiv x \pmod{p}$ en langage mathématique d'autre part on sait que d'après la propriété de congruence on peut multiplier les 2 côtés d'une congruence par une constante sans que le module change. Donc ici on aura

$a \times a^{i-1} \equiv x \times a \pmod{p} \rightarrow a^i \equiv ax \pmod{p}$ en langage informatique :

$(a^i \% p)$ et donc :

$$(a^{i \% p}) \% p$$

D'après la définition de congruence si on prend deux fois le reste de la division euclidienne d'un entier par un autre il donne la même chose donc :

$$(a^{i \% p}) \% p = (a^{i \% p})$$

Ce qui est bien le résultat souhaité.

3-Programme n°3

Un programme de décomposition en facteurs premier en utilisant le programme n°1.

```
#Programme de décomposition en facteur premier
import math
def estpremier(n):
    """
    int->bool
    estpremier(n): dit si un nombre est premier (renvoie True ou False) en
    utilisant le fait que : Si n est un nombre composé alors il a au moins un
    diviseur inférieur à racine de n."""
    if n < 7:
        if n in (2, 3, 5):
            return True
        else:
            return False
    # si n est pair et >2 (=2: cas traité ci-dessus), il ne peut pas être premier
    if n & 1 == 0:
        return False
    # autres cas
    k = 3
    r = math.sqrt(n)
    while k <= r:
        if n % k == 0:
            return False
        k += 2
    return True

assert(estpremier(5) == True)
assert(estpremier(38) == False)

def facteurpremier(n):
    """int->liste[int]
    renvoi la liste des facteurs premiers de n."""
    s = 2
    L = []

    if estpremier(n) == True:
        return n

    else:
        for s in range(2, n + 1):
            if n % s == 0:
                if estpremier(s) == True:
                    L.append(s)

    return L

assert (facteurpremier(120) == [2, 3, 5])
assert (facteurpremier(2305) == [5, 461])
assert (facteurpremier(13) == 13)
```

4-programme n°4

Un test de primalité basé sur la réciproque de théorème de Fermat :

```
programme-4.py - C:\Users\hammo\Documents\ARE-evaluation numéro-2\programme-4.py (3.5.1)
File Edit Format Run Options Window Help

def fonc2(a,n):
    """int->int
    calcul a^(n-1)% n sans calculer a^(n-1) """
    k=1
    i=0

    while i<n-1:
        k=(k*a)%n
        i=i+1
    return k

def fonc3(a,n,p):
    """int->int
    calcul a^(n-1)% n sans calculer a^((n-1)/p)%n"""
    k=1
    i=0

    while (i<(n-1)//p):
        k=(k*a)%n
        i=i+1
    return k

def pgcd(a,b):
    """int->int
    pgcd(a,b): calcul du 'Plus Grand Commun Diviseur' entre les 2 nombres entiers a et b"""
    if b==0:
        return a
    else:
        r=a%b
        return pgcd(b,r)

def reciproqueFermat(n):
    """int->bool
    la réciproque de theroem de Fermat nous dit que l'entier n est premier si:
    si il existe un a appartenant à premier à n tel que on a:
    a^(n-1)%n ==1 et a^((n-1)/p)%n!=1 ∀ p telque (n-1)%p==0
    la fonction renvoie True si le nombre est premier ,False si non."""

    n=0
    s=0
    q=0
    L=[]
    x=0
    M=[]
    e=1

    if n==1:
        return False
    elif n%2==0 and n!=2:
        return False
    elif n==2:
        return True
    else:
        while e<n:
            #On construit l'ensemble M contenant tous les a telque a est premier à n.
            if pgcd(e,n)==1:
                M.append(e)
                e=e+1

            for x in range (2,n):
                #On construit l'ensemble L contenant tous les p telque (n-1)%p==0.
                if n-1%x==0:
                    L.append(x)

            while s<len(L):
                #On verifie la premeir condition si c'est le cas on verifie la suite si non on renvoie False.
                while g<len(M):
                    if fonc3(M[g],n,L[s])!=1:
                        q=g+1
                        s=s+1
                    else:
                        return False
                g=0
                while g<len(M):
                    #On verifie la deuxiem condition on revoie True si c'est le cas ,False si non.
                    if fonc2(M[g],n)==1:
                        q=g+1
                    else:
                        return False
                g=0

    return True

assert(reciproqueFermat(22)==False)
assert(reciproqueFermat(27)==False)
assert(reciproqueFermat(361)==True)
```