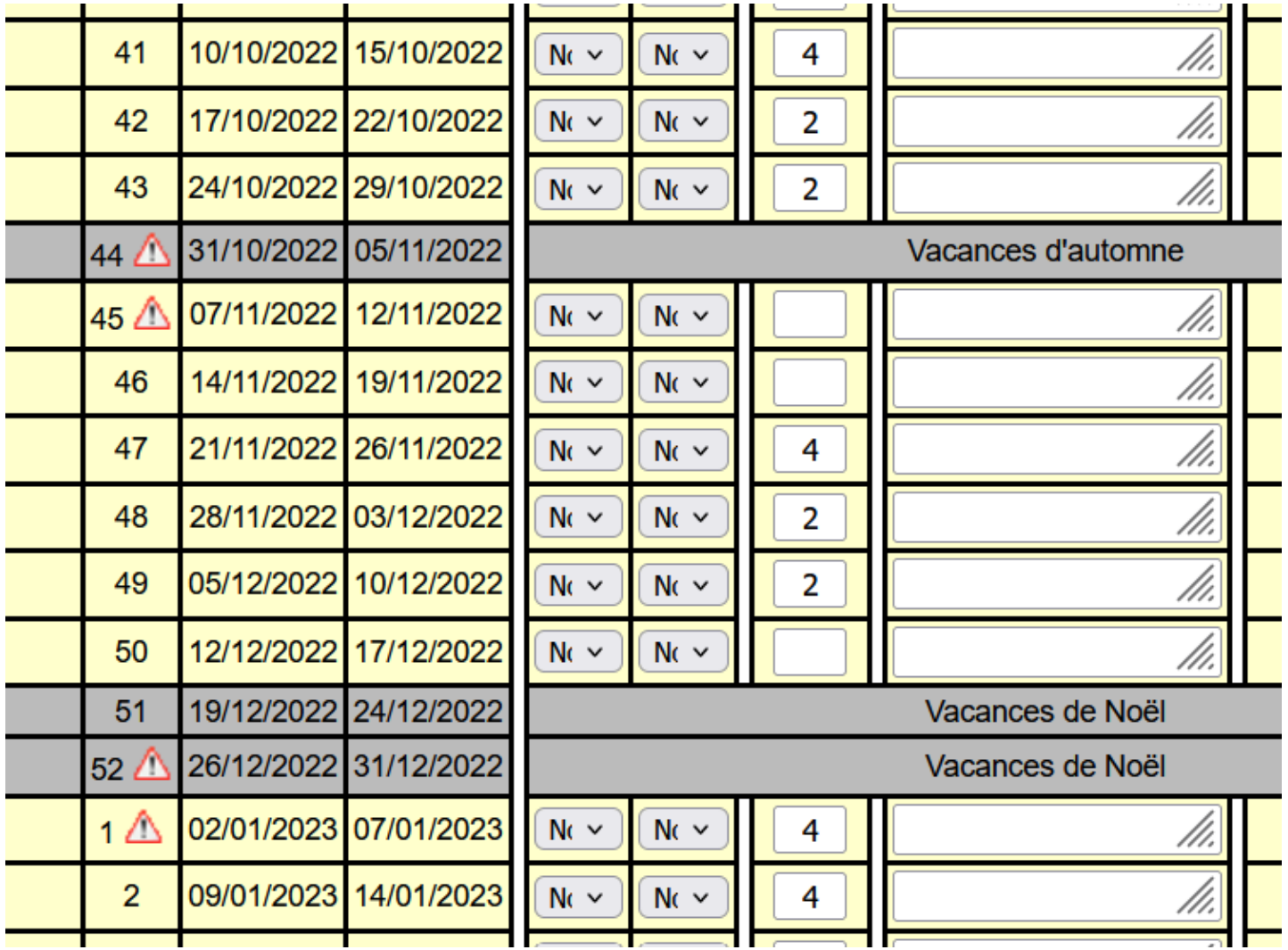
R3.09 Cryptographie

R3.09 Cryptographie



	41	10/10/2022	15/10/2022	N	N	4	
	42	17/10/2022	22/10/2022	N	N	2	
	43	24/10/2022	29/10/2022	N	N	2	
	44 ⚠	31/10/2022	05/11/2022	Vacances d'automne			
	45 ⚠	07/11/2022	12/11/2022	N	N		
	46	14/11/2022	19/11/2022	N	N		
	47	21/11/2022	26/11/2022	N	N	4	
	48	28/11/2022	03/12/2022	N	N	2	
	49	05/12/2022	10/12/2022	N	N	2	
	50	12/12/2022	17/12/2022	N	N		
	51	19/12/2022	24/12/2022	Vacances de Noël			
	52 ⚠	26/12/2022	31/12/2022	Vacances de Noël			
	1 ⚠	02/01/2023	07/01/2023	N	N	4	
	2	09/01/2023	14/01/2023	N	N	4	

Vocabulaire

Cryptologie (« science du secret »)

→**Cryptographie** (« écrire des secrets ») : vise à concevoir des systèmes sûrs pour la transmission de messages.

→**Chiffrement** : codage du message

→**Cryptogramme** : le message codé ou chiffré

→**Cryptanalyse** : vise à trouver les failles dans les systèmes de chiffrement, retrouver le message caché dans le cryptogramme

Quelques exemples historiques

Deux approches classiques de chiffrement :

- **Transposition** : l'ordre des symboles est bouleversé

- **Substitution** : les symboles du messages sont remplacés par d'autres

Chiffrement par transposition : le scytale

- Utilisé par les Spartiates (VI^{è} siècle avant JC) pour lire ou écrire des messages chiffrés.

- Bande enroulée autour d'un bâton, le message est ensuite écrit sur la longueur du bâton et le cryptogramme est obtenu en déroulant la bande.

- Pour déchiffrer le message, il suffit d'utiliser un bâton ou un cylindre de même diamètre que celui qui a été utilisé pour l'écriture du message.

Chiffrement par substitution

Remplacer les symboles du message par d'autres symboles

- **Chiffrement mono-alphabétique** (Antiquité) : chaque symbole est associé à un symbole unique.

- **Chiffrement poly-alphabétique** (Moyen Age) : une même lettre ne sera pas toujours remplacée par la même lettre, selon sa position dans le texte.

Chiffrement mono-alphabétique

Dénombrement des possibilités

Vulnérabilité

Parenthèse mathématique : dénombrements

Avec un alphabet de 26 lettres, combien a-t-on de substitutions mono-alphabétiques possibles ?

Une première façon de raisonner,

il y a :

- 26 lettres possibles pour remplacer $\boldsymbol{a}$,
- 25 pour remplacer $\boldsymbol{b}$
- …
- Il n'en reste qu'1 seule pour $\boldsymbol{z}$

$$26! = 26 \times 25 \times \cdots \times 2 \times 1$$

Parenthèse mathématique : dénombrements

Avec un alphabet de 26 lettres, combien a-t-on de substitutions mono-alphabétiques possibles ?

En utilisant un outil de dénombrement : **la permutation**
Une **permutation** est **une façon d'ordonner** les n éléments d'un ensemble.
Avec n éléments, on peut obtenir $n!$ permutations.

Ici, choisir une substitution mono-alphabétique revient à associer à l'alphabet initial, un alphabet ordonné dans un autre ordre. Cela revient à associer à l'alphabet initial, une **permutation** de ce dernier. Il y a donc $26!$ chiffrements possibles.

VULNERABILITÉ DE LA SUBSTITUTION MONO-ALPHABÉTIQUE

La cryptanalyse de ce type de chiffrement repose sur l'analyse des fréquences des symboles qui sont les mêmes dans le message initial et dans son chiffrement.

Lettres les plus fréquentes

E	A	I	S	T	N	R	U	L	O	D	M	P	C	V	Q
15,9%	9,4%	8,4%	7,9%	7,3%	7,2%	6,5%	6,2%	5,3%	5,1%	3,4%	3,2%	2,9%	2,6%	2,2%	1,1%

Digrammes les plus fréquents

ES	LE	EN	DE	RE	NT	ON	TE	ER	SE
3,15 %	2,46 %	2,42 %	2,15 %	2,09 %	1,97 %	1,64 %	1,63 %	1,63 %	1,55 %

Source Wikipédia

https://www.apprendre-en-ligne.net/crypto/stat/francais.html

VULNERABILITÉ DE LA SUBSTITUTION MONO-ALPHABÉTIQUE

La cryptanalyse repose sur l'analyse des fréquences des symboles qui sont les mêmes dans le message initial et dans son chiffrement.

```
from collections import Counter
c=Counter(cryptogramme)

freq=c.most_common(10)
```

	D	M	P	C	V	Q
5,1%	3,4%	3,2%	2,9%	2,6%	2,2%	1,1%

	TE	ER	SE
	1,63 %	1,63 %	1,55 %

Source Wikipédia

```
[(' ', 47), ('a', 43), ('g', 22), ('w', 19), ('s', 16), ('l', 15), ('u', 15), ('p', 15), ('r', 14), ('v', 8)]
```

Remarque préliminaire : codage des lettres

a → 0, b → 1, … , z → 25

TRAVAIL SOUS PYTHON

1. Quelle structure de données Python pourrions-nous utiliser pour numéroter les lettres de l'alphabet ?

2. Créer une fonction **corresp(alpha)** qui renvoie l'objet ci-dessus en prenant en paramètre une chaîne de caractères listant toutes les lettres de l'alphabet.

   Exemple de paramètre : alpha = « abcdefghijklmnopqrstuvwxyz »
   Remarque : en Python, les chaînes de caractères sont itérables

3. À partir de l'objet créé précédemment, expliquez comment récupérer le code à partir d'une lettre et inversement, comment récupérer la lettre à partir du code. Créer une ou des fonctions, si nécessaire.

TRAVAIL SOUS PYT

1. Quelle structure de données Py
de l'alphabet ?

2. Créer une fonction **corresp(alph**
paramètre une chaîne de caract

Exemple de paramètre : alpha =
Remarque : en Python, les chaîn

3. À partir de l'objet créé précéder
à partir d'une lettre et inversem
Créer une ou des fonctions, si n

```
alpha="abcdefghijklmnopqrstuvwxyz"

def corresp(alpha):
    dic={}
    for i in range(len(alpha)):
        dic[alpha[i]]=i
    return dic

dic = corresp(alpha)

# dic --> {'a': 0, 'b': 1, 'c': 2 ...  'z': 25}

# Obtenir un code à partir d'une lettre ('a' par
exemple)
dic['a']    # --> 0

# Obtenir une lettre à partir de son code
def retrouvelettre(code):
    for lettre in dic:
        if dic[lettre] == code:
            return lettre

retrouvelettre(2)       # --> 'c'
```

CHIFFREMENT DE CESAR

SUBSTITUTION MONO-ALPHABÉTIQUE

CHIFFREMENT DE CÉSAR

Pour chiffrer ses messages, Jules César décalait chaque lettre de **trois unités** : a donnait d, b donnait e, c donnait f…, x donnait a, y donnait b, z donnait c.

Par exemple, pour le message « prenezuntaxi » :

p r e n e z u n t a x i
↓↓↓↓↓↓↓↓↓↓↓↓↓
s u h q h c x q w d a l

Le cryptogramme est « suhghcxqwdal »

CHIFFREMENT DE CÉSAR

Quelle opération utiliser pour faire ce décalage, si on utilise des nombres entiers pour représenter des lettres ?

écalait chaque lettre de **trois**
ait f…, x donnait a, y donnait b,

untaxi » :

p r e n e z u n t a x i
↓↓↓↓↓↓↓↓↓↓↓↓↓
s u h q h c x q w d a l

Le cryptogramme est « suhghcxqwdal »

Parenthèse d'arithmétique

Congruence dans Z

Définition : deux entiers a et b sont **congrus modulo n** s'ils ont le même reste par la division par n.

Notation : $\boldsymbol{a \equiv b\ [n]}$

Parenthèse d'arithmétique

Congruence dans Z

Définition : deux entiers a et b sont **congrus modulo n** s'ils ont le même reste par la division par n.

Notation : $\boldsymbol{a \equiv b\ [n]}$

$\boldsymbol{a}$ et $\boldsymbol{b}$ « congrus modulo $\boldsymbol{n}$ »

$\Leftrightarrow$

$(\boldsymbol{a - b})$ multiple de $\boldsymbol{n}$

$\Leftrightarrow$

$\boldsymbol{a - b = kn,}$ **avec** $\boldsymbol{k} \in \mathbb{N}$

$\Leftrightarrow$

$\boldsymbol{a = b + kn,}$ **avec** $\boldsymbol{k} \in \mathbb{N}$

La relation « congruence modulo n »

La relation « congruence modulo n » est une relation entre les entiers relatifs, il s'agit donc d'un graphe dont les sommets sont les éléments de $\mathbb{Z}$

Quelles sont les propriétés de ce graphe (ou de cette relation) ?

Symétrie

Si $\boldsymbol{a}$ et $\boldsymbol{b}$ ont le même reste par la division par $\boldsymbol{n}$, alors $\boldsymbol{b}$ et $\boldsymbol{a}$ ont le même reste par la division par $\boldsymbol{n}$

Malgré une notation « asymétrique », **la congruence est une relation symétrique :**

$$\boldsymbol{a \equiv b\ [n] \Leftrightarrow b \equiv a [n]}$$

C'est la raison pour laquelle on la note parfois $\equiv_n$

$\boldsymbol{7 \equiv 4[3]}$ s'écrit aussi $\boldsymbol{7 \equiv_3 4}$ ou $\boldsymbol{4 \equiv_3 7}$

La relation « congruence modulo n »

Réflexivité

De façon évidente, $\boldsymbol{a}$ à le même reste que $\boldsymbol{a}$ dans la division euclidienne par $\boldsymbol{n}$: $\boldsymbol{a} \equiv \boldsymbol{a}\ [\boldsymbol{n}]$

La congruence est donc une relation réflexive

Transitivité

Si $\boldsymbol{a}$ et $\boldsymbol{b}$ ont le même reste par la division par $\boldsymbol{n}$ et si Si $\boldsymbol{b}$ et $\boldsymbol{c}$ ont le même reste par la division par $\boldsymbol{n}$ alors Si $\boldsymbol{a}$ et $\boldsymbol{c}$ ont le même reste par la division par $\boldsymbol{n}$:

$$\left.\begin{array}{l} \boldsymbol{a} \equiv \boldsymbol{b}\ [\boldsymbol{n}] \\ \boldsymbol{b} \equiv \boldsymbol{c}\ [\boldsymbol{n}] \end{array}\right\} \Longrightarrow \boldsymbol{a} \equiv \boldsymbol{c}\ [\boldsymbol{n}]$$

La congruence est donc une relation transitive

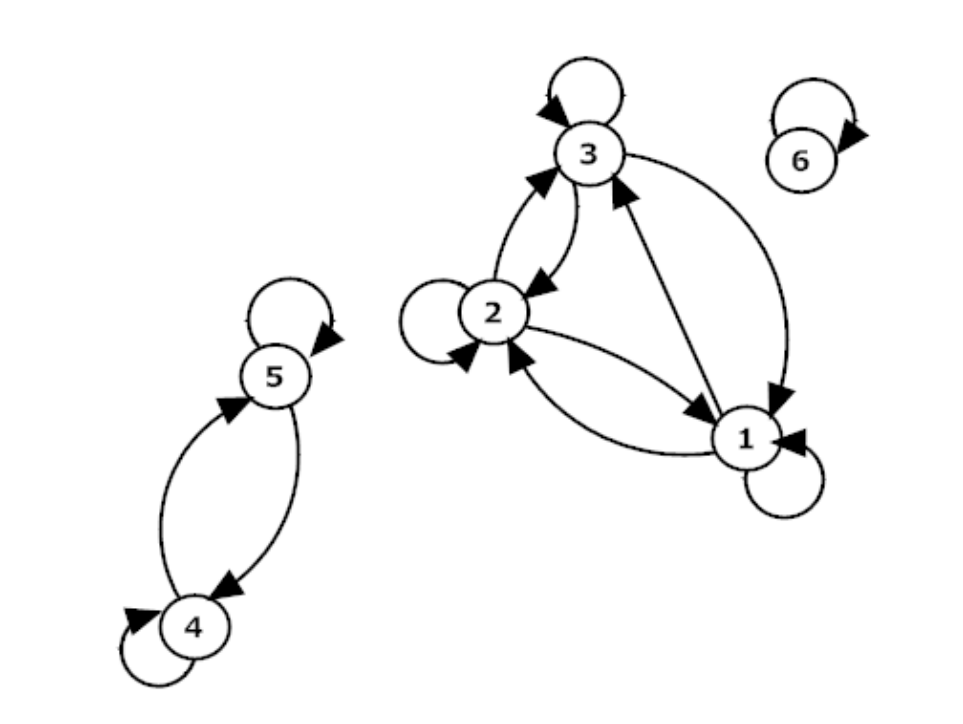
La relation « congruence modulo n »

Bilan, la congruence est une relation :

- **Réflexive**
- **Symétrique**
- **Transitive**

⇒ la congruence est une relation d'équivalence

Relation d’équivalence ?



Elle se caractérise par un graphe constitué de **sous-graphes complets**.

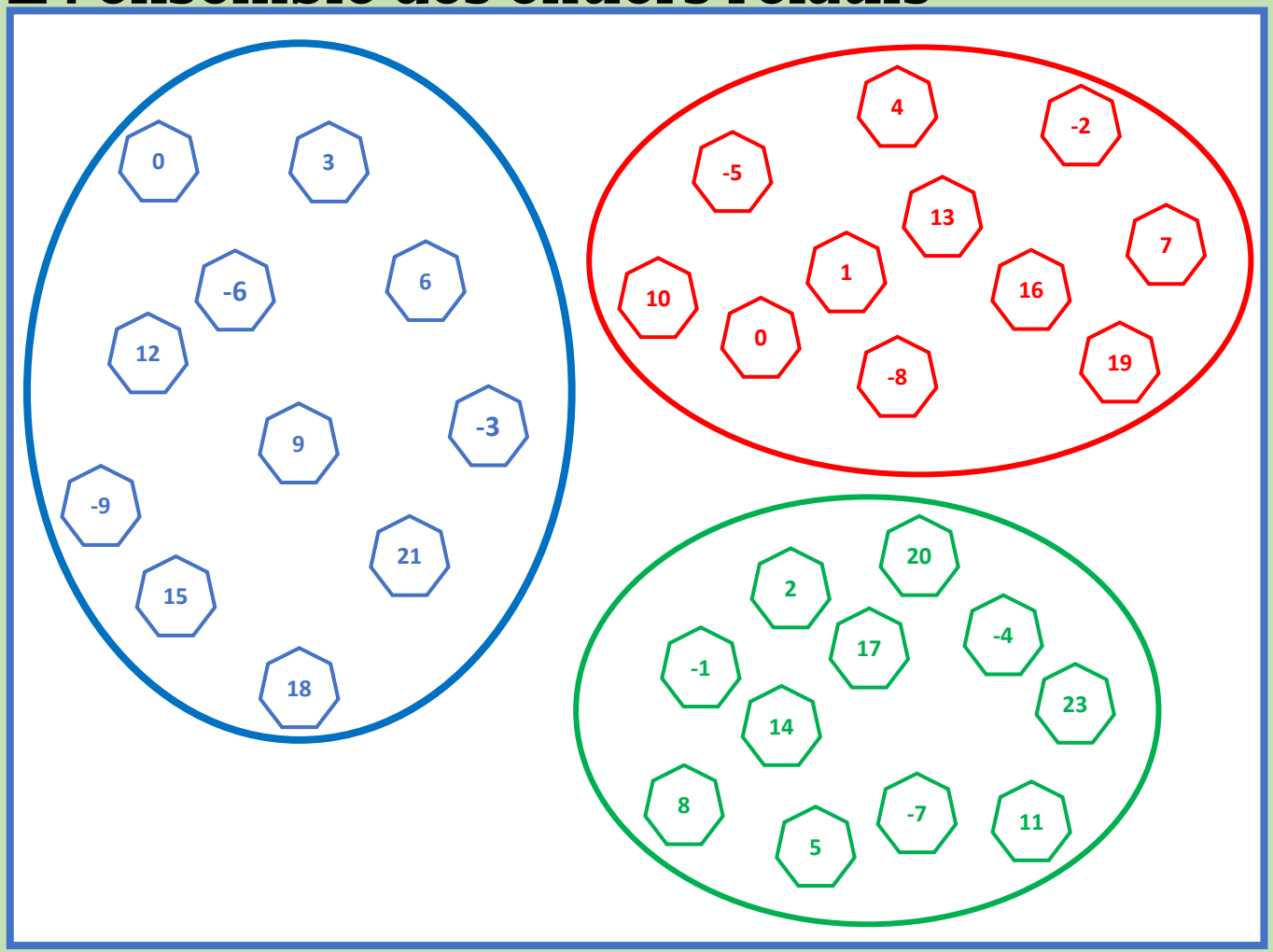
Ce qui permet de regrouper les sommets dans des sous-ensembles appelés **classes d’équivalences**. Ici : $\{1, 2, 3\}, \{4, 5\}, \{6\}$

L’ensemble des classes d’équivalence est une **partition** de l’ensemble des sommets S et est appelé **ensemble quotient** de la relation R et se note S/R

$$S/R = \{1, 2, 3\}, \{4, 5\}, \{6\}$$

$\mathbb{Z}/3\mathbb{Z}$: ensemble quotient de la congruence modulo 3

$\mathbb{Z}$: ensemble des entiers relatifs



Trois classes d'équivalence :

Les entiers relatifs congrus à 0 modulo3 :

$$\dot{0} = \{\dots -9, -6, -3, 0, 3, 6, 9, \dots\}$$

Les entiers relatifs congrus à 1 modulo3 :

$$\dot{1} = \{\dots -8, -5, -2, 1, 4, 7, 10, \dots\}$$

Les entiers relatifs congrus à 2 modulo3 :

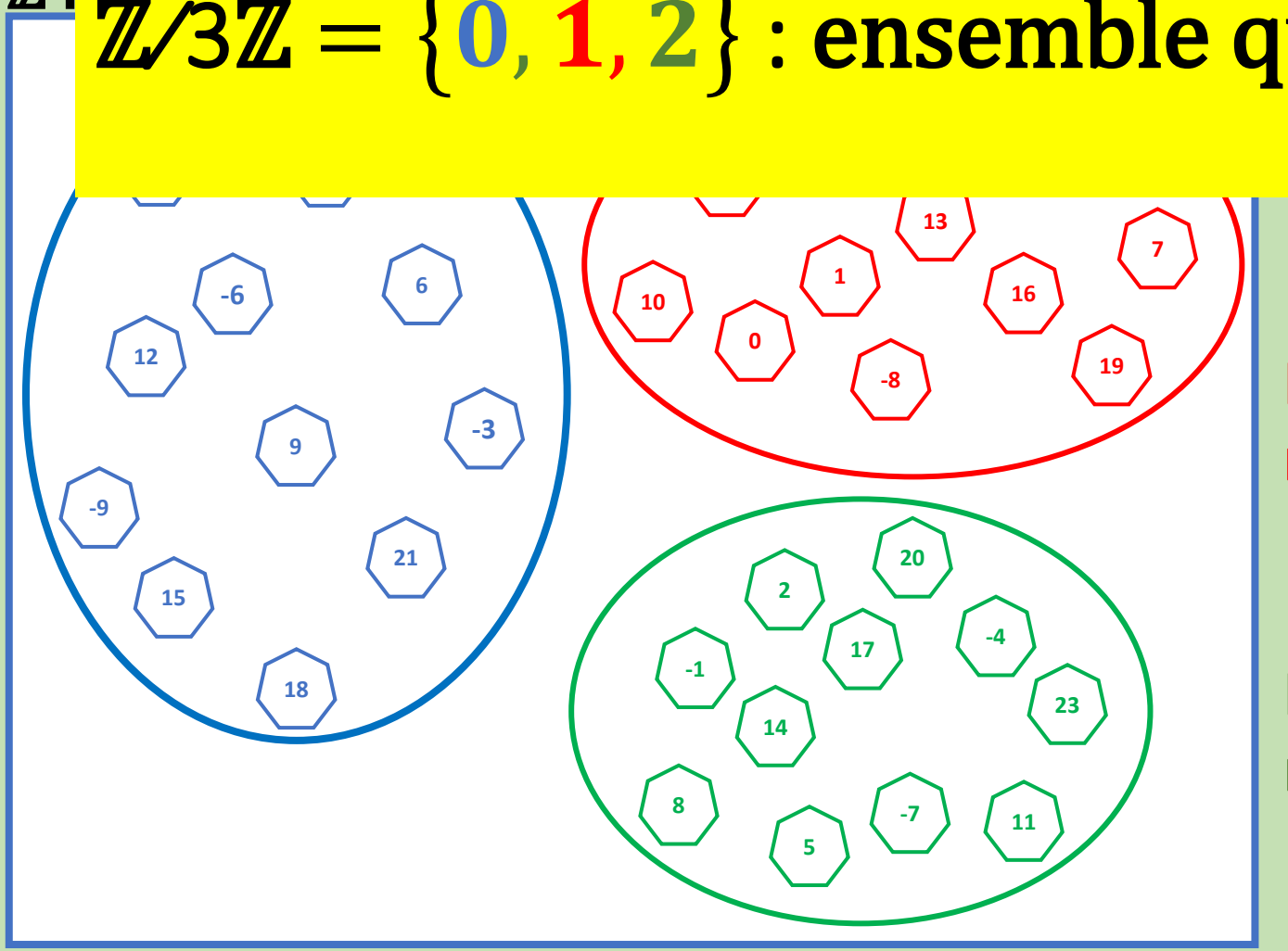
$$\dot{0} = \{\dots -7, -4, -1, 2, 5, 8, 11, \dots\}$$

$\mathbb{Z}/3\mathbb{Z}$: ensemble quotient de la congruence modulo 3

$\mathbb{Z}$:

$\mathbb{Z}/3\mathbb{Z} = \{\dot{0}, \dot{1}, \dot{2}\}$: ensemble quotient

Il pourrait se noter $\mathbb{Z}/\equiv_3$!



$\dot{0} = \{\dots -9, -6, -3, 0, 3, 6, 9, \dots\}$

Les entiers relatifs congrus à 1 modulo3 :

$\dot{1} = \{\dots -8, -5, -2, 1, 4, 7, 10, \dots\}$

Les entiers relatifs congrus à 2 modulo3 :

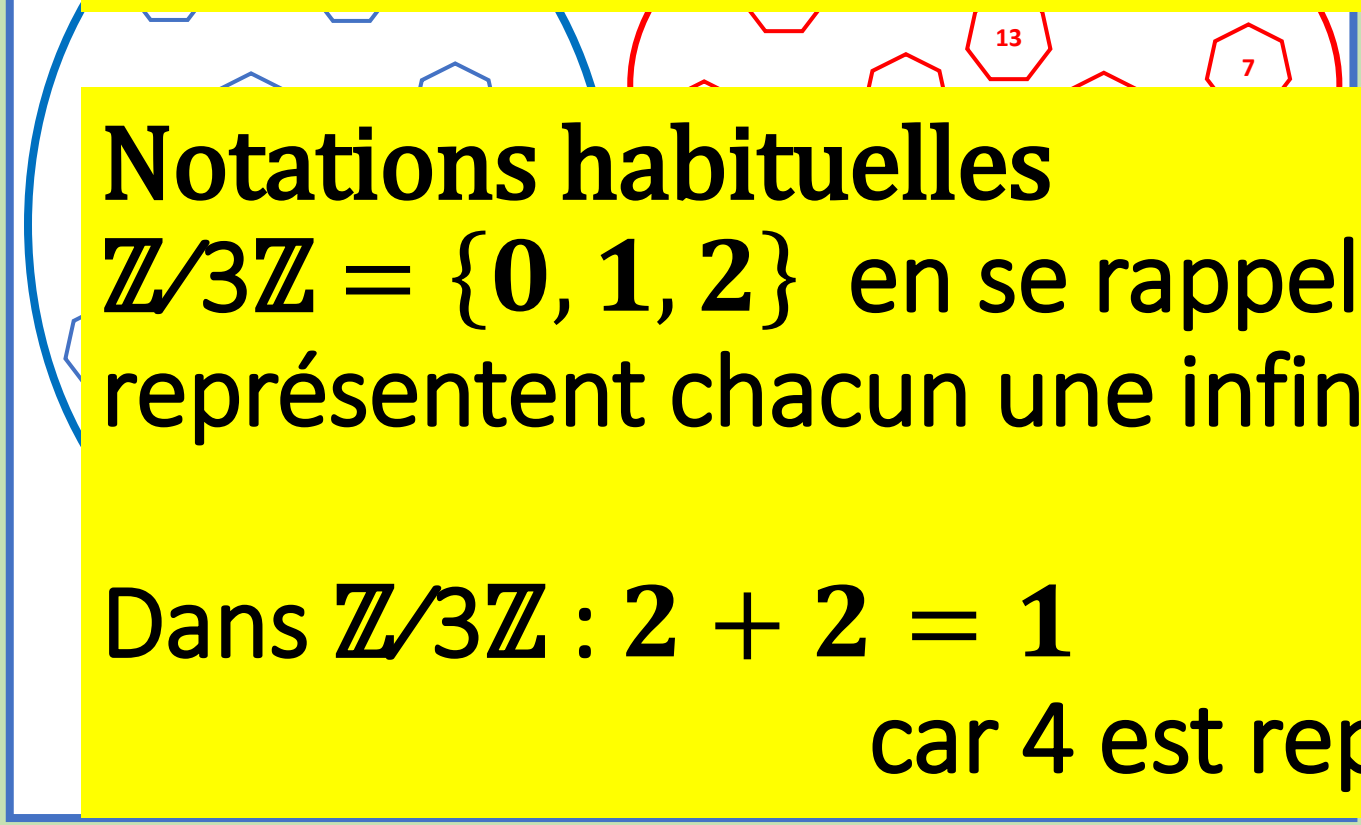
$\dot{2} = \{\dots -7, -4, -1, 2, 5, 8, 11, \dots\}$

$\mathbb{Z}/3\mathbb{Z}$: ensemble quotient de la congruence modulo 3

$\mathbb{Z}$:

$\mathbb{Z}/3\mathbb{Z} = \{\dot{0}, \dot{1}, \dot{2}\}$: ensemble quotient

Il pourrait se noter $\mathbb{Z}/\equiv_3$!



$\dot{0} = \{\ldots -9, -6, -3, 0, 3, 6, 9, \ldots\}$

Notations habituelles

$\mathbb{Z}/3\mathbb{Z} = \{0, 1, 2\}$ en se rappelant que $0, 1, 2$ représentent chacun une infinité d'entiers.

Dans $\mathbb{Z}/3\mathbb{Z}$: $2 + 2 = 1$

car 4 est représenté par 1 dans $\mathbb{Z}/3\mathbb{Z}$

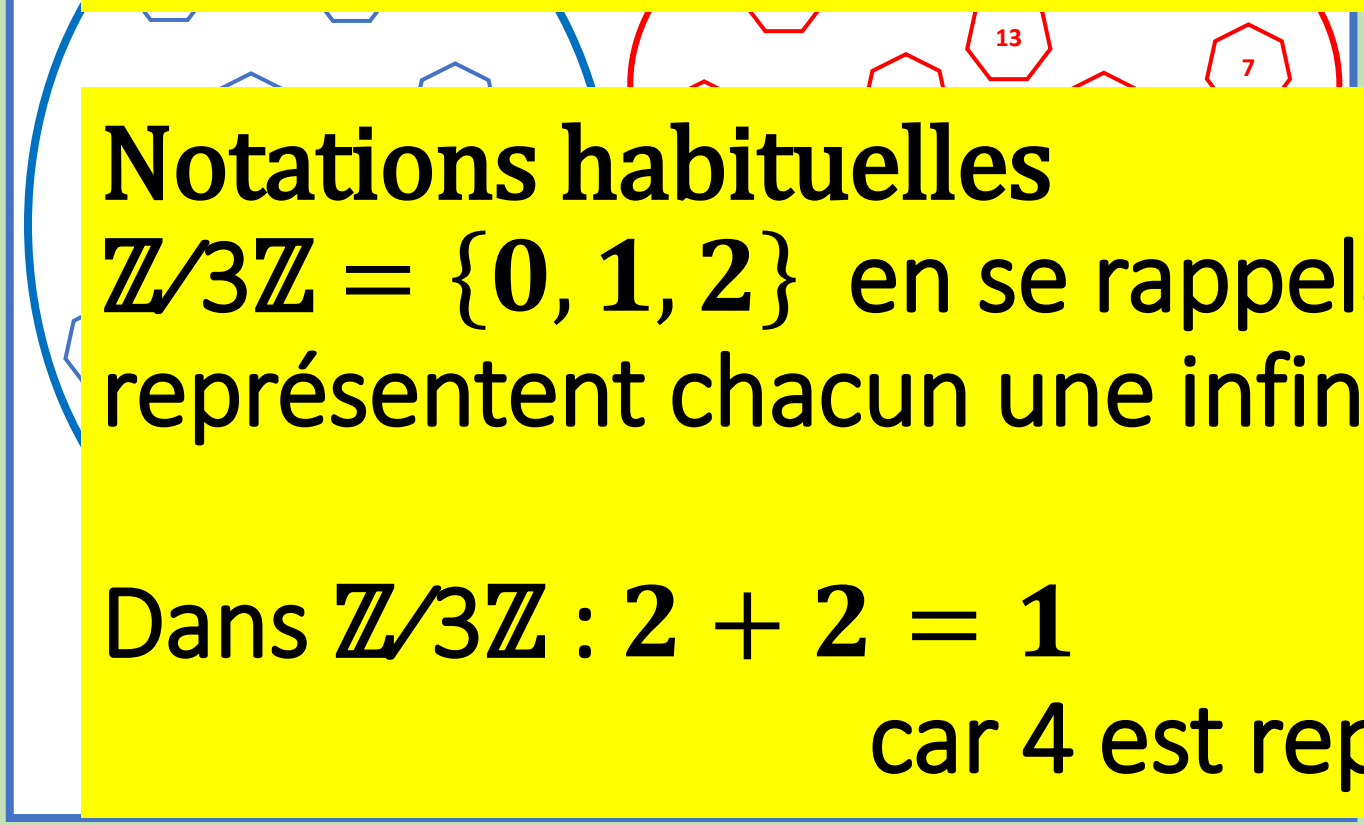
$\mathbb{Z}/3\mathbb{Z}$: ensemble quotient de la congruence modulo 3

$\mathbb{Z}$:

$\mathbb{Z}/3\mathbb{Z} = \{\dot{0}, \dot{1}, \dot{2}\}$: ensemble quotient

Il pourrait se noter $\mathbb{Z}/\equiv_3$!

$\dot{0} = \{\ldots -9, -6, -3, 0, 3, 6, 9, \ldots\}$



Notations habituelles

$\mathbb{Z}/3\mathbb{Z} = \{0, 1, 2\}$ en se rappelant que $0, 1, 2$ représentent chacun une infinité d'entiers.

Dans $\mathbb{Z}/3\mathbb{Z}$: $2 + 2 = 1$

car 4 est représenté par 1 dans $\mathbb{Z}/3\mathbb{Z}$

Généralisation : $\mathbb{Z}/n\mathbb{Z}$

$$\mathbb{Z} \,/\, n\mathbb{Z} \;= \{0, 1, \ldots, n-1\}$$

Remarque importante en Python :

Pour faire des calcul dans $\mathbb{Z}/n\mathbb{Z}$ en Python, il suffit de leur appliquer systématiquement l'opérateur $\%n$ qui permet d'obtenir le **reste** de la division euclidienne par n et ainsi un nombre congru au résultat et compris entre 0 et $n-1$, donc élément de $\mathbb{Z}/n\mathbb{Z}$.

Revenons au chiffrement de César

CHIFFREMENT DE CÉSAR « AMÉLIORÉ » AVEC CLÉ DE CHIFFREMENT.

On décale les lettres de **c lettre(s).**

Le chiffrement dépend ici d'un paramètre **c**, qui doit être gardé secret et qui sera appelé **clé de chiffrement**.

Comme la clé sera utilisée à la fois pour coder et décoder le message, on parlera ici de **cryptographie symétrique à clé secrète**.

Python

1. Programmer les fonctions suivantes :

✓ **chiffrecesar(message,cle)** prenant en entrée le message et la clé et renvoyant le cryptogramme

✓ **dechiffrecesar(cryptogramme,cle)** prenant en entrée le cryptogramme et la clé, et renvoyant le message

2. **Cryptanalyse** : Décoder le cryptogramme suivant obtenu par un chiffrement de César avec une clé inconnue (à trouver !) :

'vccvgivjjrjlicruvkvekvkflkvjcvjzuvvjulscfeuzevkjvgrigzccvivektvcrwzklevafczvwcvliurejcvtzvcuyzmvirmrekhlvcvgivdzvigvkrcvvewlkivkfdsvvcrmzvzccvrmrzkivdzjvjferidvurejjfetrsrjvkivgiverzkjriflkvurezvcgveertcrwvvtrirszev'

```
# fonction codageCesar(message,cle). En entrée, le message à coder et la clé (décalage). En sortie, le cryptogramme
def codageCesar(message,cle):
    code=''
    for lettre in message:
        decalage=(dic[lettre]+cle)%26
        lettrecodee=trouvecle(dic,decalage)
        code=code+lettrecodee
    return code

codageCesar('bonjour',3)          # --> 'erqmrxu'

def dechiffrecesar(cryptogramme,cle):
    return chiffrecesar(cryptogramme,-cle)

dechiffrecesar('erqmrxu',3)       # --> 'bonjour'

#  Cryptanalyse de César à clé
cryptogramme='vccvgivjjrjlicruvkvekvkflkvjcvjzuvvjulscfeuzevkjvgrigzccvivektvcrwzklevafczvwcvliurejcvtzvcuyzmvirmrekhlvcvgivdzvigvkrcvve
wlkivkfdsvvcrmzvzccvrmrzkivdzjvjferidvurejjfetrsrjvkivgiverzkjriflkvurezvcgveertcrwvvtrirszev'

# détection de la clé par analyse fréquentielle
from collections import Counter
c=Counter(cryptogramme)
freq=c.most_common(5)             # --> [('v', 45), ('r', 22), ('c', 19), ('e', 16), ('i', 15)]

# On suppose que e est codé v, la clé est donc dic['v']-dic['e']
dechiffrecesar(cryptogramme, dic['v']-dic['e'])

#'ellepressasurladetentetouteslesideesdublondinetseparpillerentcelafitunejoliefleurdanslecieldhiveravantquelepremierpetaleenfutretom
beelavieilleavaitremisesonarmedanssoncabasetreprenaitsaroutedanielpennaclafeecarabine'
```

CHIFFREMENT AFFINE

SUBSTITUTION MONO-ALPHABÉTIQUE

Chiffrement affine

la **clé est un couple d'entier (a,b)** : chaque lettre de rang $\boldsymbol{i}$ est remplacée par la lettre de rang $\boldsymbol{j = ai + b[26]}$

Chiffrement affine

la **clé est un couple d'entier (a,b)** : chaque lettre de rang i est remplacée par la lettre de rang $\boldsymbol{j = ai + b[26]}$

Mais est-on sûr que l'opération soit **réversible** ?

Chiffrement affine

la **clé est un couple d'entier (a,b)** : chaque lettre de rang i est remplacée par la lettre de rang $\boldsymbol{j = ai + b[26]}$

Mais est-on sûr que l'opération soit **réversible** ?

$$\boldsymbol{j = ai + b[26] \Longrightarrow ai = j - b[26] \Longrightarrow i = a^{-1}(j - b)[26]}$$

Chiffrement affine

la **clé est un couple d'entier (a,b)** : chaque lettre de rang i est remplacée par la lettre de rang $\boldsymbol{j = ai + b[26]}$

Mais est-on sûr que l'opération soit **réversible** ?

$$\boldsymbol{j = ai + b[26] \Longrightarrow ai = j - b[26] \Longrightarrow i = a^{-1}(j - b)[26]}$$

a est-il toujours inversible modulo 26, c'est-à-dire dans $\mathbb{Z}/26\mathbb{Z}$?
Comment calcule-t-on a^{-1}?

PARENTHÈSE D'ARITHMÉTIQUE

PGCD

Principe de l'algorithme d'Euclide

Théorème de Bezout (identité de Bezout)

Nombres premiers entre eux

Inversibilité dans Z/nZ

PARENTHÈSE D'ARITHMÉTIQUE

PGCD

Le PGCD de deux entiers relatifs a et b est le « plus grand commun diviseur de a et b », c'est-à-dire le plus grand entier d divisant à la fois a et b.

Principe de l'algorithme d'Euclide

Théorème de Bezout (identité de Bezout)

Nombres premiers entre eux

Inversibilité dans Z/nZ

PARENTHÈSE D'ARITHMÉTIQUE

PGCD

Le PGCD de deux entiers relatifs a et b est le « plus grand commun diviseur de a et b », c'est-à-dire le plus grand entier d divisant à la fois a et b.

Principe de l'algorithme d'Euclide

Si q et r sont respectivement le quotient et le reste de la division euclidienne de n par m (n≥m) alors : $\boldsymbol{PGCD(n,m) = PGCD(m,r)}$

Théorème de Bezout (identité de Bezout)

Nombres premiers entre eux

Inversibilité dans Z/nZ

PARENTHÈSE D'ARITHMÉTIQUE

PGCD

Le PGCD de deux entiers relatifs a et b est le « plus grand commun diviseur de a et b », c'est-à-dire le plus grand entier d divisant à la fois a et b.

Principe de l'algorithme d'Euclide

Si q et r sont respectivement le quotient et le reste de la division euclidienne de n par m (n≥m) alors : $\boldsymbol{PGCD(n,m) = PGCD(m,r)}$

Théorème de Bezout (identité de Bezout)

Soient $\boldsymbol{n}$ et $\boldsymbol{m}$ deux entiers naturels.

Il existe deux entiers relatifs, u et v de Z, tels que : $\boldsymbol{PGCD(n,m) = u \times n + v \times m}$

Les entiers $\boldsymbol{u}$ et $\boldsymbol{v}$ sont appelés coefficients de Bezout.

Nombres premiers entre eux

Inversibilité dans Z⁄nZ

PARENTHÈSE D'ARITHMÉTIQUE

Nombres premiers entre eux

Deux entiers n et m sont premiers entre eux s'ils ont pour seuls diviseurs communs 1 et -1.

Inversibilité dans Z/nZ

Soient $\boldsymbol{p}$ et $\boldsymbol{n}$ deux entiers ($\boldsymbol{p} \leq \boldsymbol{n}$)

Les propositions suivantes sont équivalentes :

- $\boldsymbol{p}$ est inversible dans $\mathbb{Z}/n\mathbb{Z}$
- $\boldsymbol{n}$ et $\boldsymbol{p}$ sont premiers entre eux
- $\boldsymbol{PGCD(n,p) = 1}$
- Identité de Bezout : $\exists u, v \in Z$ tels que $\boldsymbol{nu + pv = 1}$

Algorithme d'Euclide étendu

Déterminer $\boldsymbol{pgcd(165, 72)}$ et les coefficients de Bezout $\boldsymbol{u}$ et $\boldsymbol{v}$ vérifiant :

$$\boldsymbol{pgcd(165, 72) = 165 \times u + 72 \times v}$$

Calcul des coefficients de Bezout

$$\boldsymbol{pgcd(165, 72) = 165 \times u + 72 \times v}$$

Principe : reprendre l'algorithme d'Euclide, en exprimant, à chaque étape le reste comme combinaison linéaire de 165 et 72.
Le dernier reste non nul correspond au pgcd et nous donnera donc l'identité de Bezout et ses coefficients

Calcul des coefficients de Bezout

$$\boldsymbol{pgcd(165, 72) = 165 \times u + 72 \times v}$$

Division euclidienne (algorithme d'Euclide)	**Reste =$165 \times u + 72 \times v$**
$165 = 2 \times 72 + 21$	$\mathbf{21 = 165 - 2 \times 72}$
$72 = 3 \times 21 + 9$	$9 = 72 - 3 \times \mathbf{21}$ $9 = 72 - 3 \times (\mathbf{165 - 2 \times 72})$ $\mathbf{9 = -3 \times 165 + 7 \times 72}$
$21 = 2 \times 9 + 3$	$3 = \mathbf{21} - 2 \times \mathbf{9}$ $3 = \mathbf{165 - 2 \times 72} - 2 \times (\mathbf{-3 \times 165 + 7 \times 72})$ $\mathbf{3 = 7 \times 165 - 16 \times 72}$
$9 = 3 \times 3 + 3$	$\boldsymbol{pgcd(165, 72) = 3}$ (dernier reste non nul) **Identité de Bezout : $\boldsymbol{pgcd(165, 72) = 7 \times 165 - 16 \times 72}$**

Opérations sur les lignes		r : reste	u (coefficient de 165)	v (coefficient de 72)	Division euclienne	q : quotient
Initialisation	L_0	165	1	0		
	L_1	72	0	1	$165 = 2 \times 72 + 21$ $21 = 165 - 2 \times 72$	2
$L_2 = L_0 - 2L_1$		21	1	-2	$72 = 3 \times 21 + 9$ $9 = 72 - 3 \times 21$	3
$L_3 = L_1 - 3L_2$		9	-3	7	$21 = 2 \times 9 + 3$ $3 = 21 - 2 \times 9$	2
$L_4 = L_2 - 2L_3$		3	7	-16	$9 = 3 \times 3 + 0$ Reste nul, c'est fini	

Exercice1

1. Euclide étendu

Écrire une fonction **`pgcd_euclide_etendu(n,m)`** prenant en paramètres deux entiers **`n`** et **`m`**, et renvoyant le tuple **`(pgcd, u, v)`** dans lequel :

- pgcd=pgcd(n,m)
- u et v sont les coefficients de Bezout dans l'égalité : $\boldsymbol{pgcd = u \times n + v \times m}$

2. Inverse modulaire

En utilisant la fonction **`pgcd_euclide_etendu(n,m)`**, écrire en Python une fonction **`inversemod(nb,mod)`** prenant en entrée deux entiers **`nb`** et **`mod`** et renvoyant l'inverse modulaire de nb quand celui-ci existe.

Exercice2

1. Si on utilise un chiffrement affine sur un alphabet de 26 lettres, combien de a-t-on de clés possibles ?
2. Programmer les fonctions suivantes :

- **`chiffreaffine(message,a,b)`** prenant en entrée le message et la clé (a,b) et renvoyant le cryptogramme
- **`dechiffreaffine(cryptogramme,a,b)`** prenant en entrée le cryptogramme et la clé (a,b) et renvoyant le message.

Exercice 2 : cryptanalyse chiffrement affine

'lqdmadtfkahuhqutadnkxxutesdstqutrqmadtfkalsrpqumqdtmq
psstnawulsfswrpulsxkatmlshsfmstladsqtwkmrnsfsmaudtqdtsd
kdrpamyaadtfkamsedamqxpkddsavmqdmfusdrkafmqmmskufd
umafyakuwqdgsfestqutadtfkalsnkxxutesyauuwrpuyaspsekdzkf
tbfftkpiusdxupxkpsnkxxut'