Gymnase de Burier

1OS 2022 – 2023

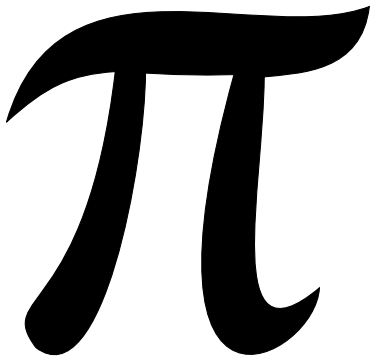


Table des matières

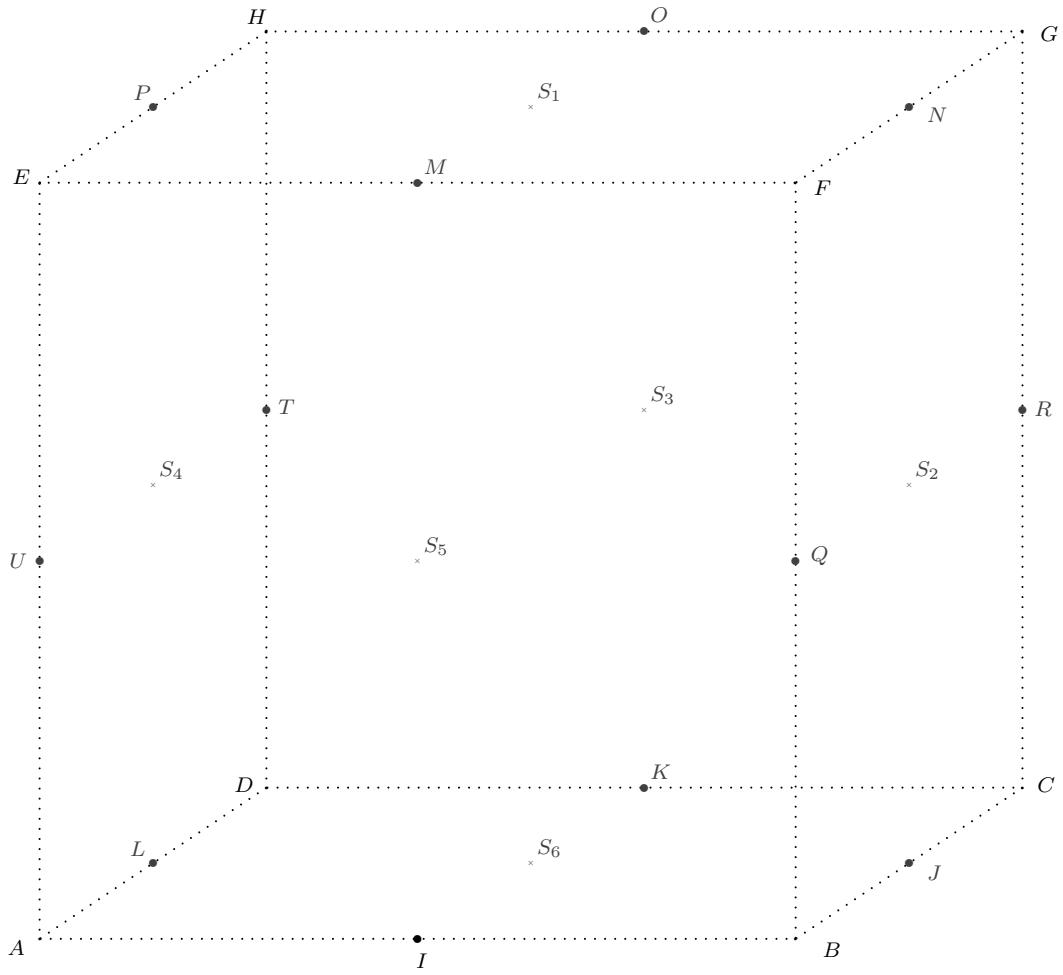
Chapitre 1

Géométrie

1.1 Tracés dans le cube

Dans le cube $ABCDEFGH$, on note :

- I le milieu de AB
- J le milieu de BC
- K le milieu de CD
- L le milieu de AD
- M le milieu de EF
- N le milieu de FG
- O le milieu de GH
- P le milieu de EH
- R le milieu de CG
- T le milieu de DH
- U le milieu de AE
- Q le milieu de BF
- S_1 le milieu de la face $EFGH$
- S_2 le milieu de la face $BCGF$
- S_3 le milieu de la face $CGHD$
- S_4 le milieu de la face $ADHE$
- S_5 le milieu de la face $ABFE$
- S_6 le milieu de la face $ABCD$



Les constructions se font sur les pages suivantes en tenant compte de la visibilité.

1.1.1 Dans le cube $ABCDEFGH$ représenter les plans $PNJL$ et $HGBA$.

1.1.2 Représenter les plans $PNCD$ et $UQRT$.

1.1.3 Notons par exemple X_{MF} le milieu de MF.
Représenter les plans $EHCB$ et $X_{MF}X_{OG}X_{KC}X_{IB}$.

1.1.4 Représenter les plans $EHCB$ et $HGBA$.

1.1.5 Représenter le triangle EHJ et le quadrilatère $S_1S_4S_6S_2$.

1.1.6 Représenter le triangle S_1BC et le plan $EHRQ$.

1.1.7 Représenter les triangles PNS_6 et AOC.

1.1.8 Représenter les triangles PNS_6 et EOC.

1.1.9 Dans le cube $ABCDEFGH$ représenter la pyramide $ABCDS_1$ et le plan $UQRT$.

1.1.10 Représenter la pyramide $ABCDS_1$ et le plan $IJNM$.

1.1.11 Représenter la pyramide $ABCDS_1$ et le plan $UQGH$.

1.1.12 Représenter la pyramide $IJKLS_1$ et le plan $BCHE$.

1.1.13 Représenter la pyramide $IJKLS_1$ et le plan $ABRT$.

1.1.14 Représenter l'octaèdre $S_1S_2S_3S_4S_5S_6$ et le plan $BDHF$.

1.1.15 Représenter l'octaèdre $S_1S_2S_3S_4S_5S_6$ et le plan $UQGH$.

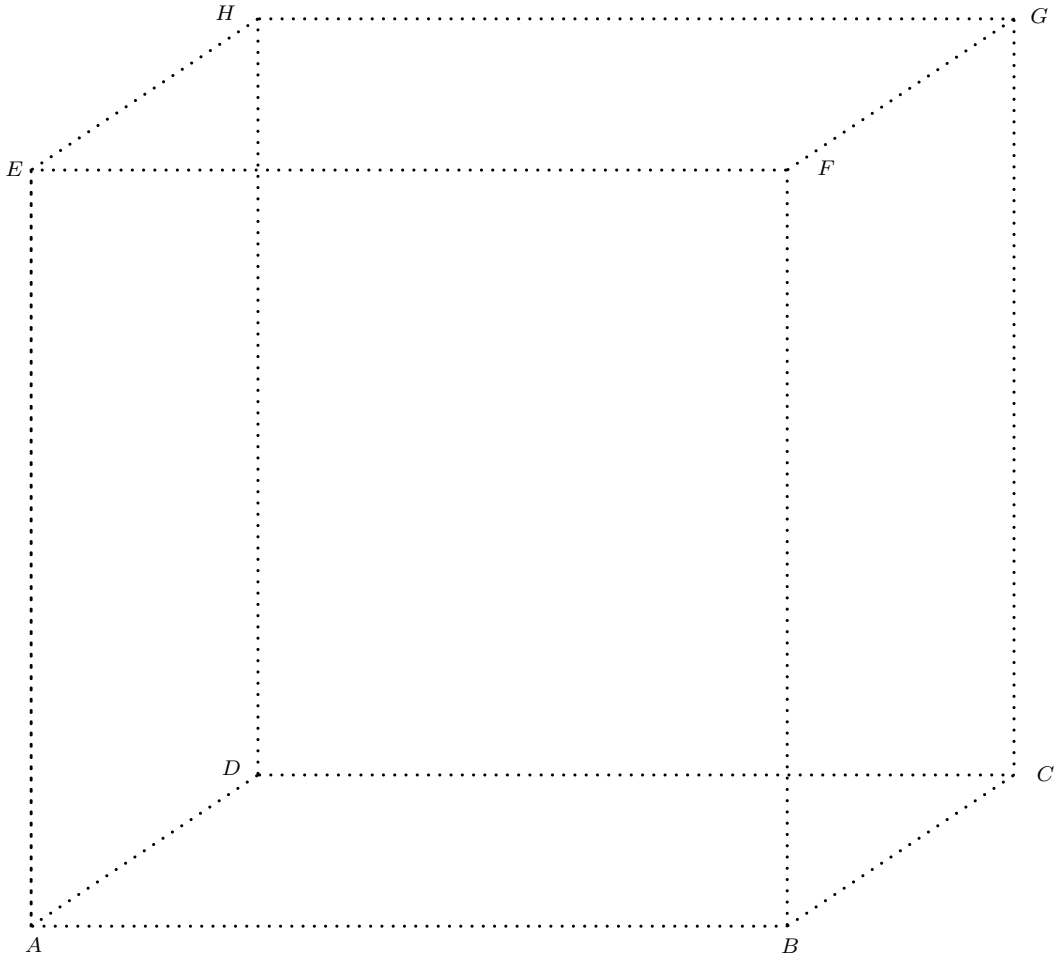
1.1.16 Représenter l'octaèdre $S_1S_2S_3S_4S_5S_6$ et le plan $ABGH$.

1.1.17 Représenter l'octaèdre $S_1S_2S_3S_4S_5S_6$ et le plan UPO.

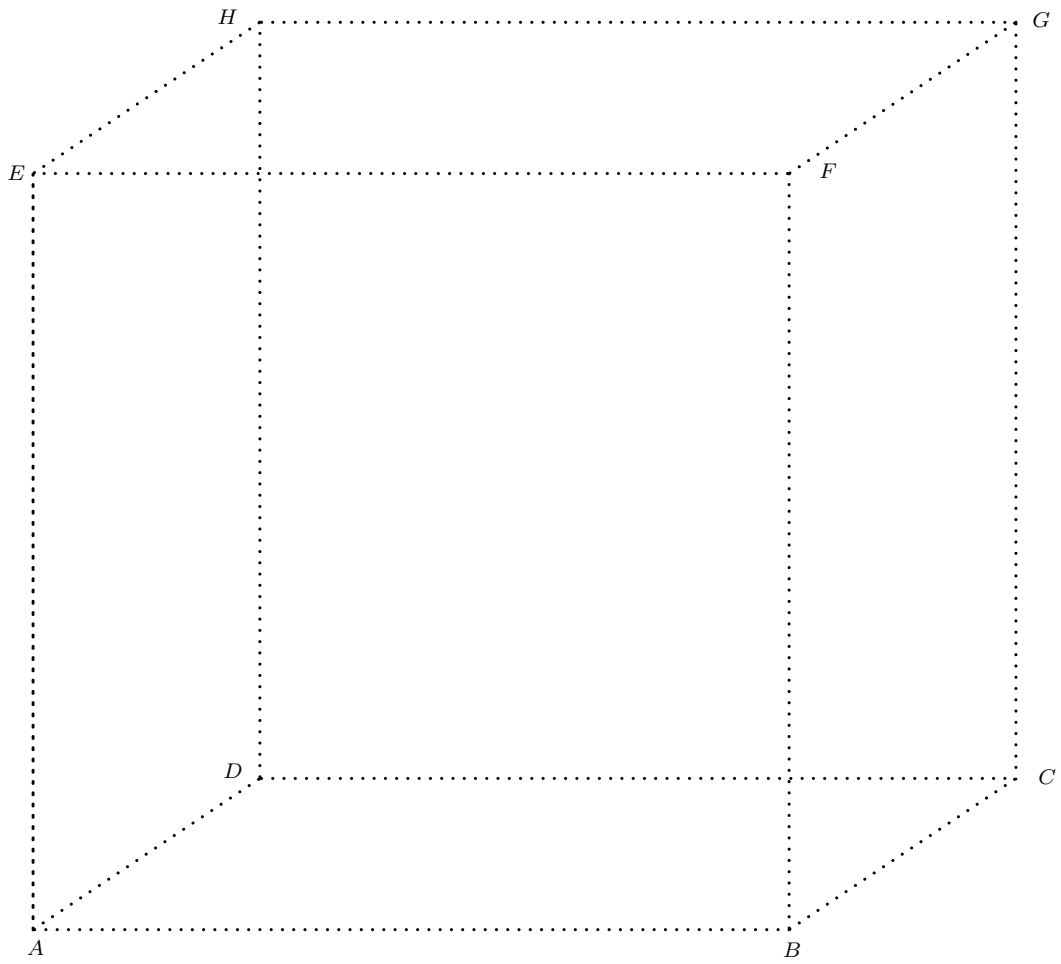
1.1.18 Soit Y sur AB tel que $YB = 2.5$ cm. Soit X sur DC tel que $XC = 2.5$ cm. Soit Z sur HG tel que $HZ = 1.0$ cm.
Représenter l'octaèdre $S_1S_2S_3S_4S_5S_6$ et le plan XYZ.

1.1.19 Représenter l'intersection des pyramides $ABCDH$ et $IJKLM$.

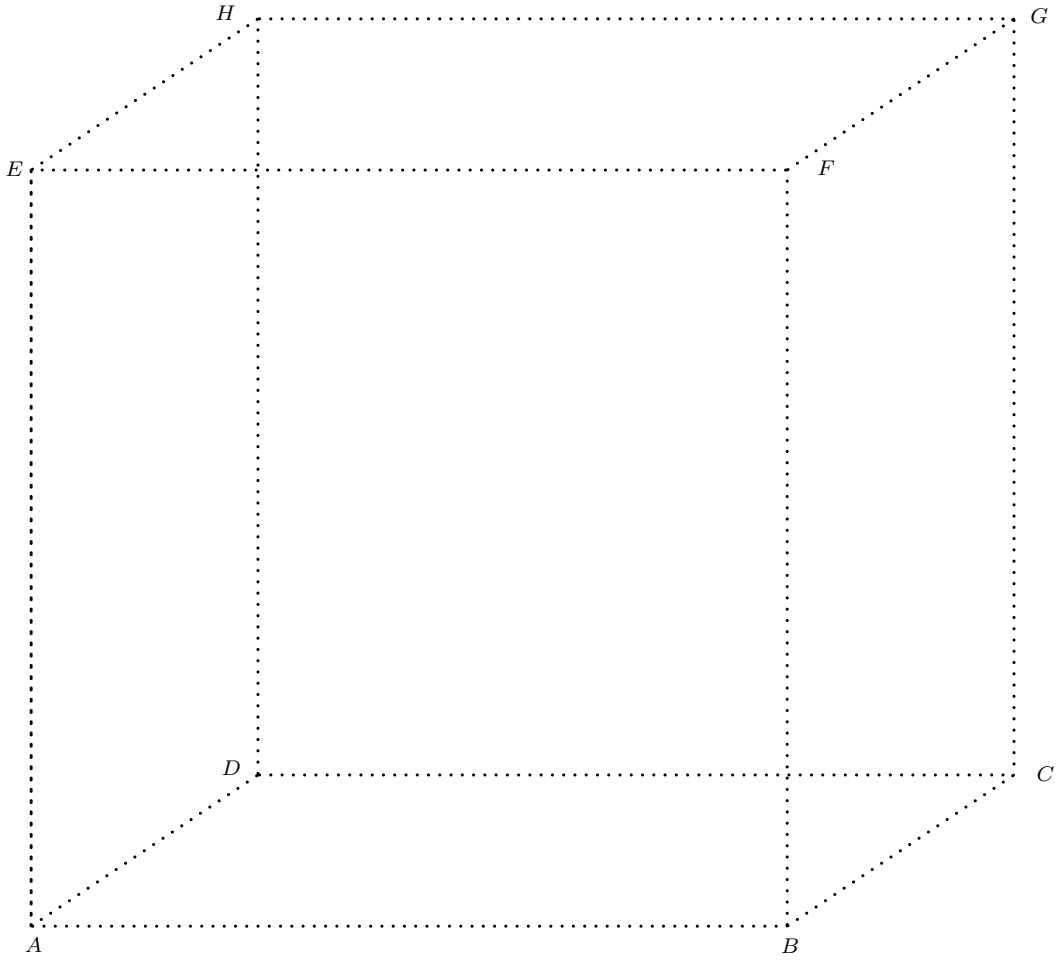
1.1.1



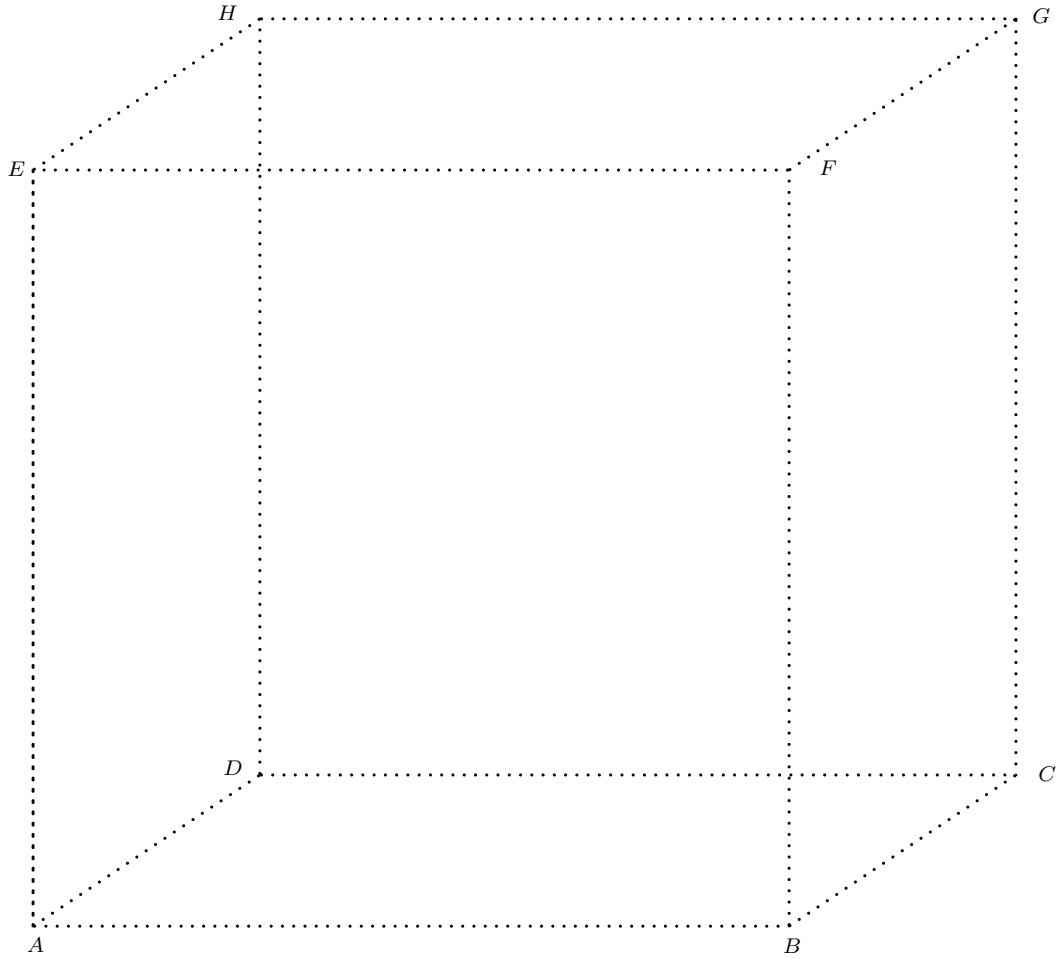
1.1.2



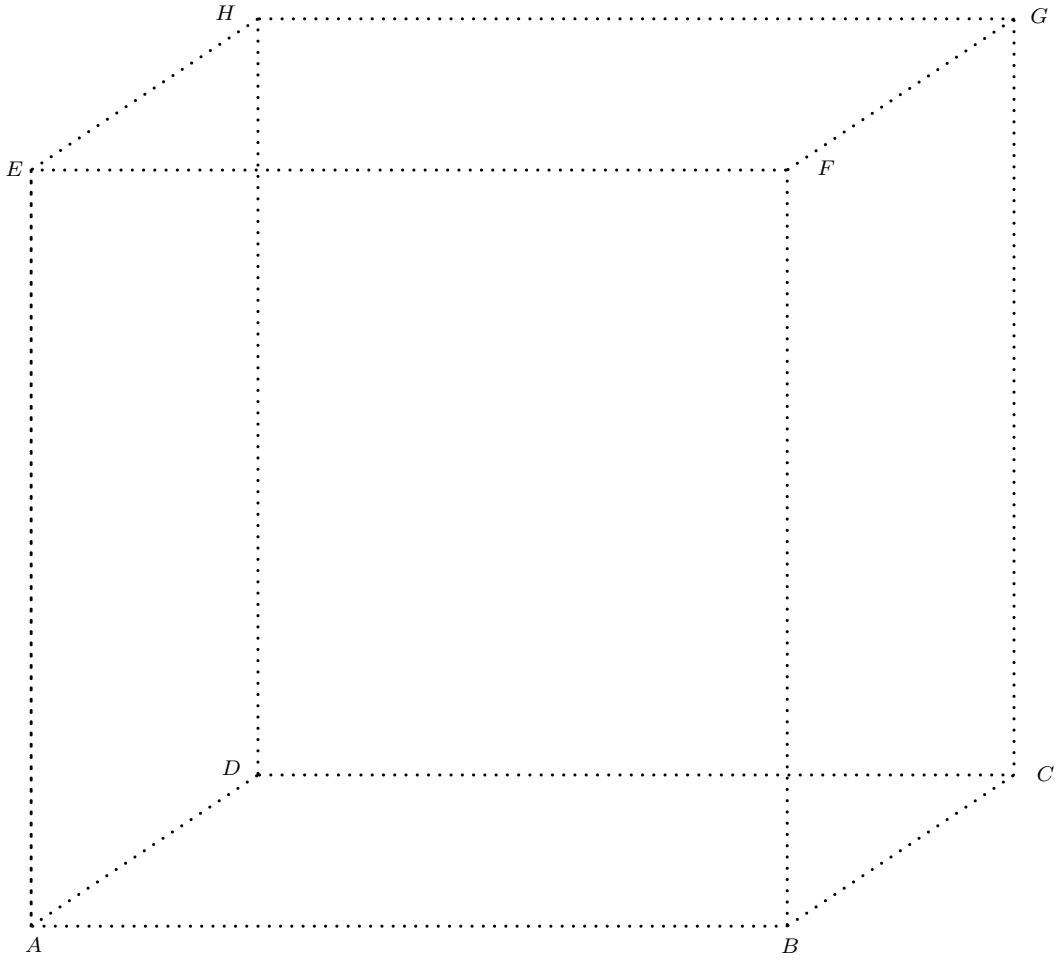
1.1.3



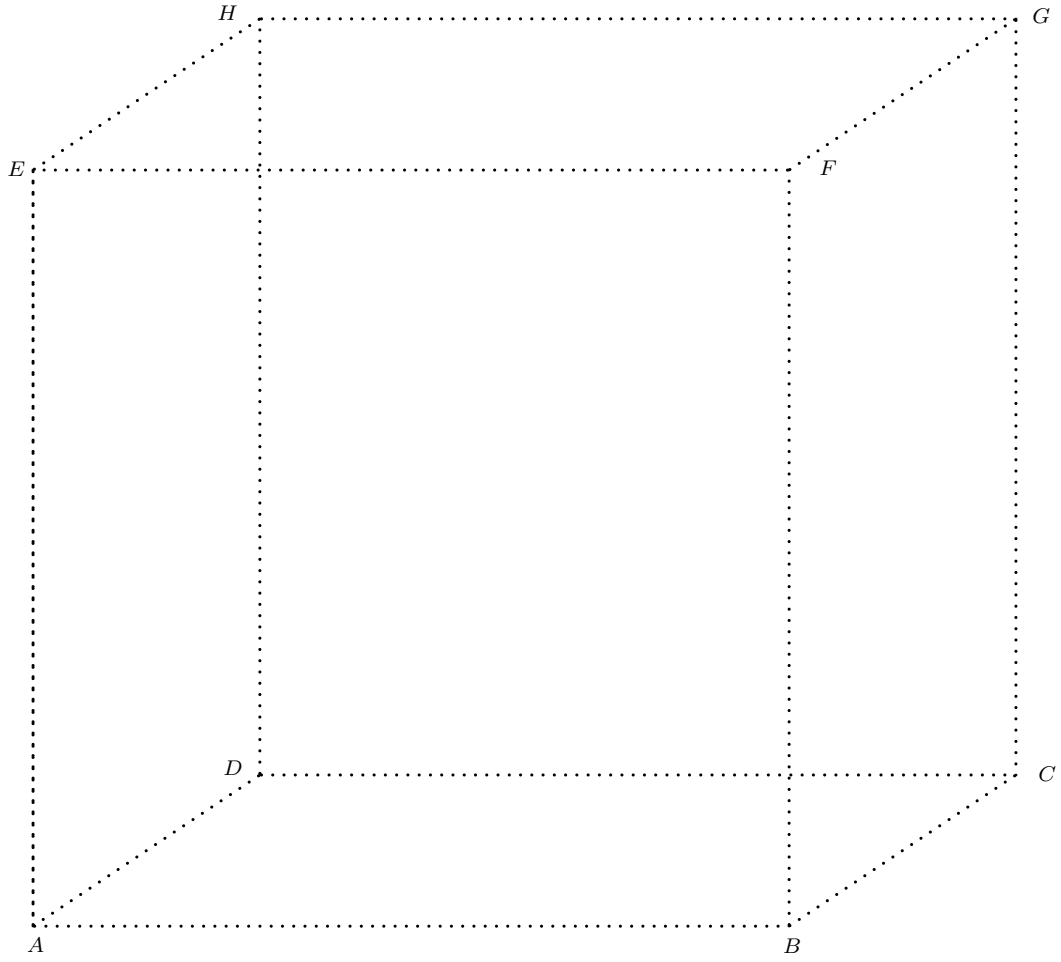
1.1.4



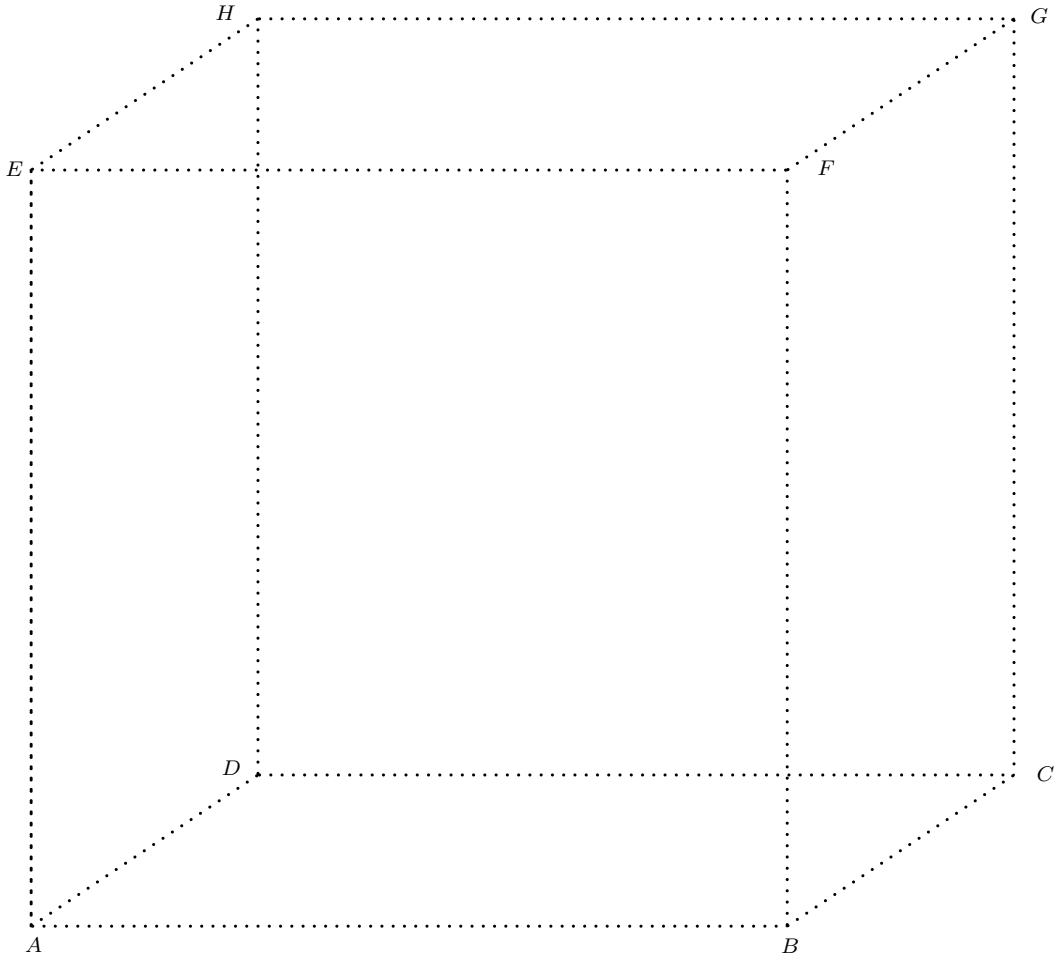
1.1.5



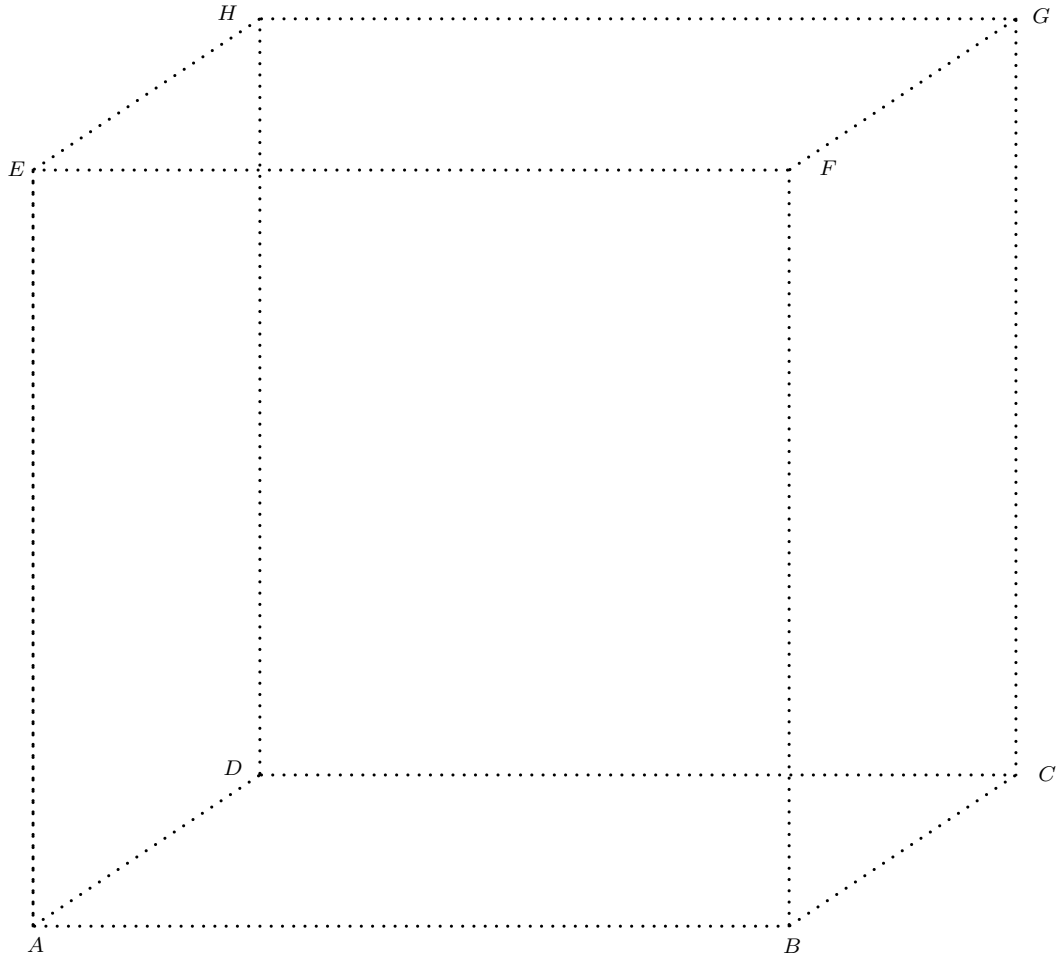
1.1.6



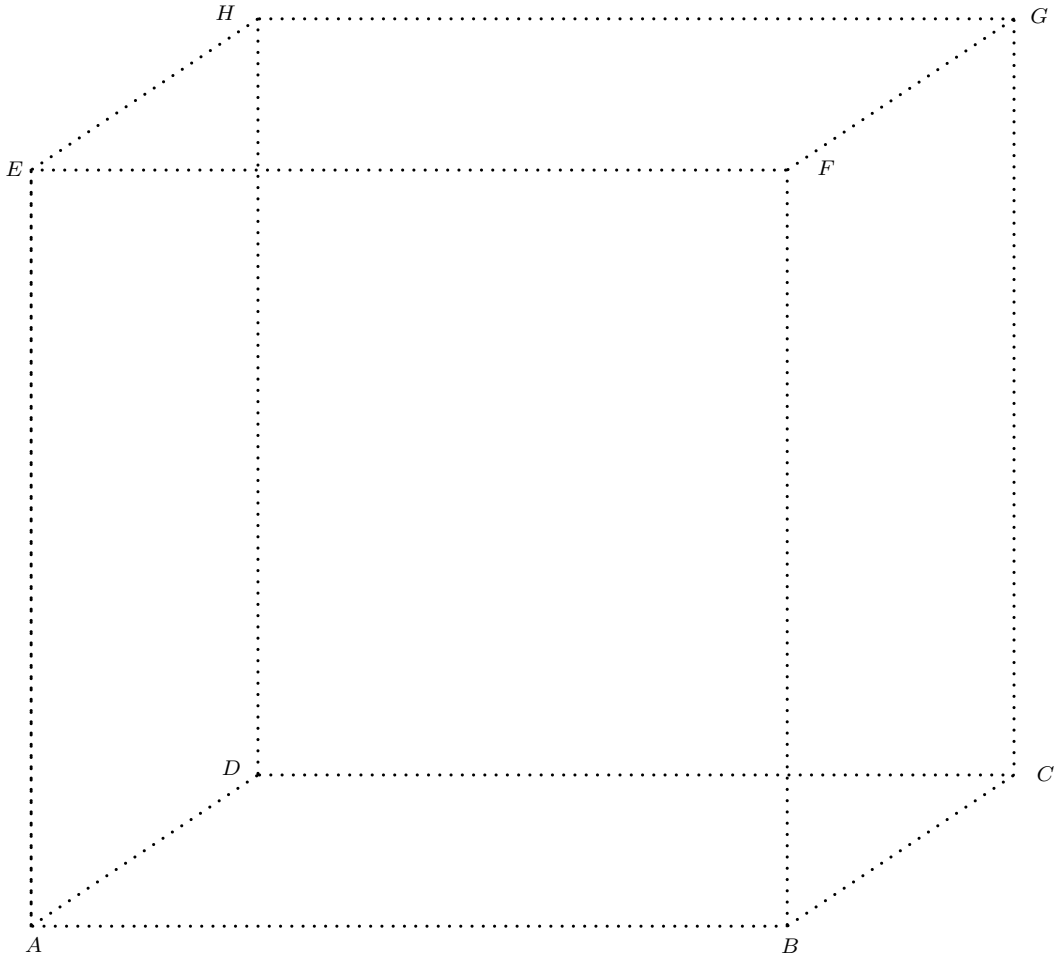
1.1.7



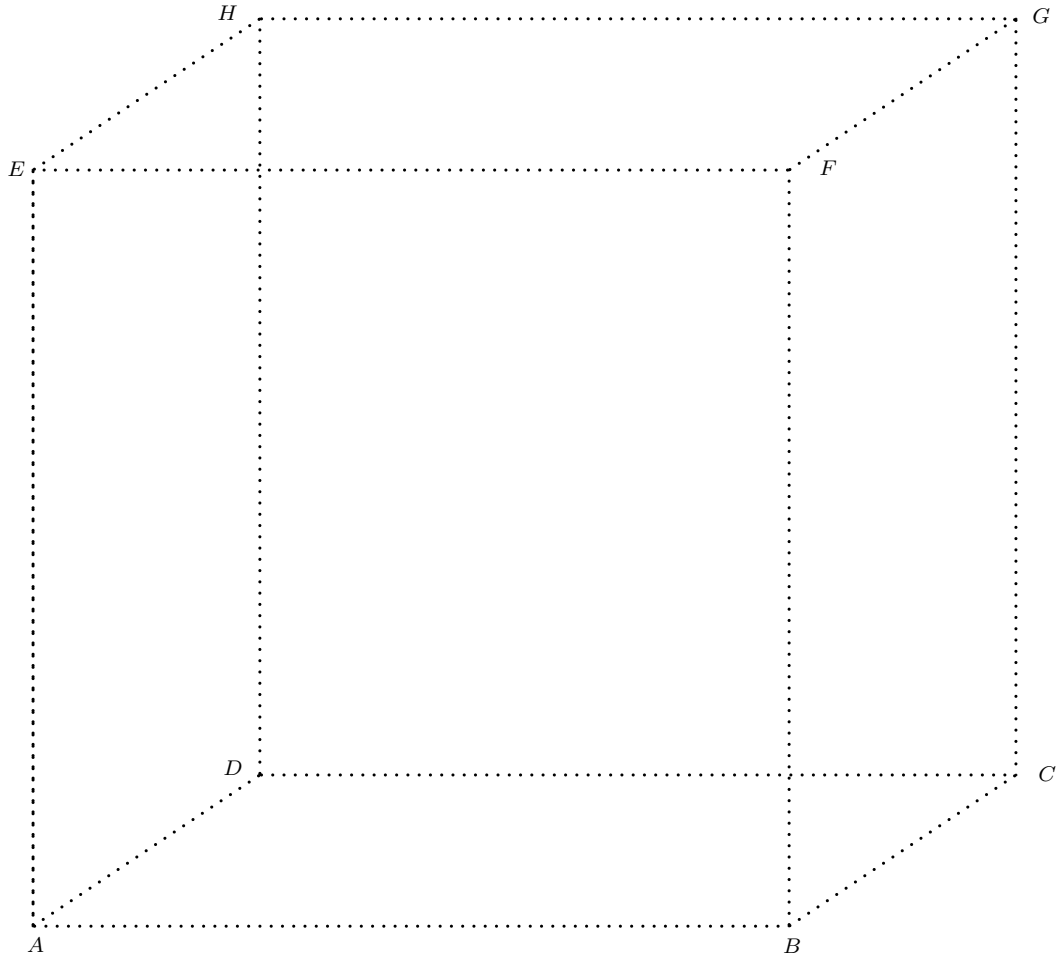
1.1.8



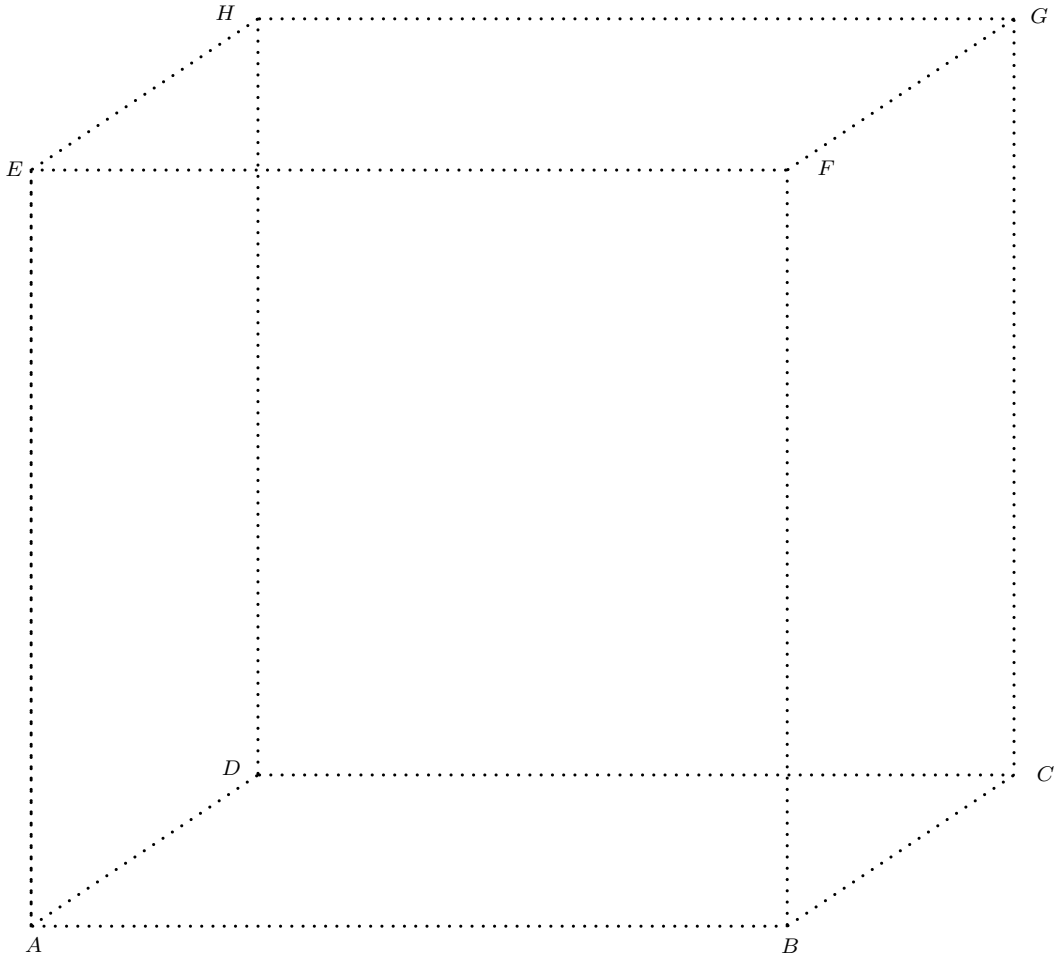
1.1.9



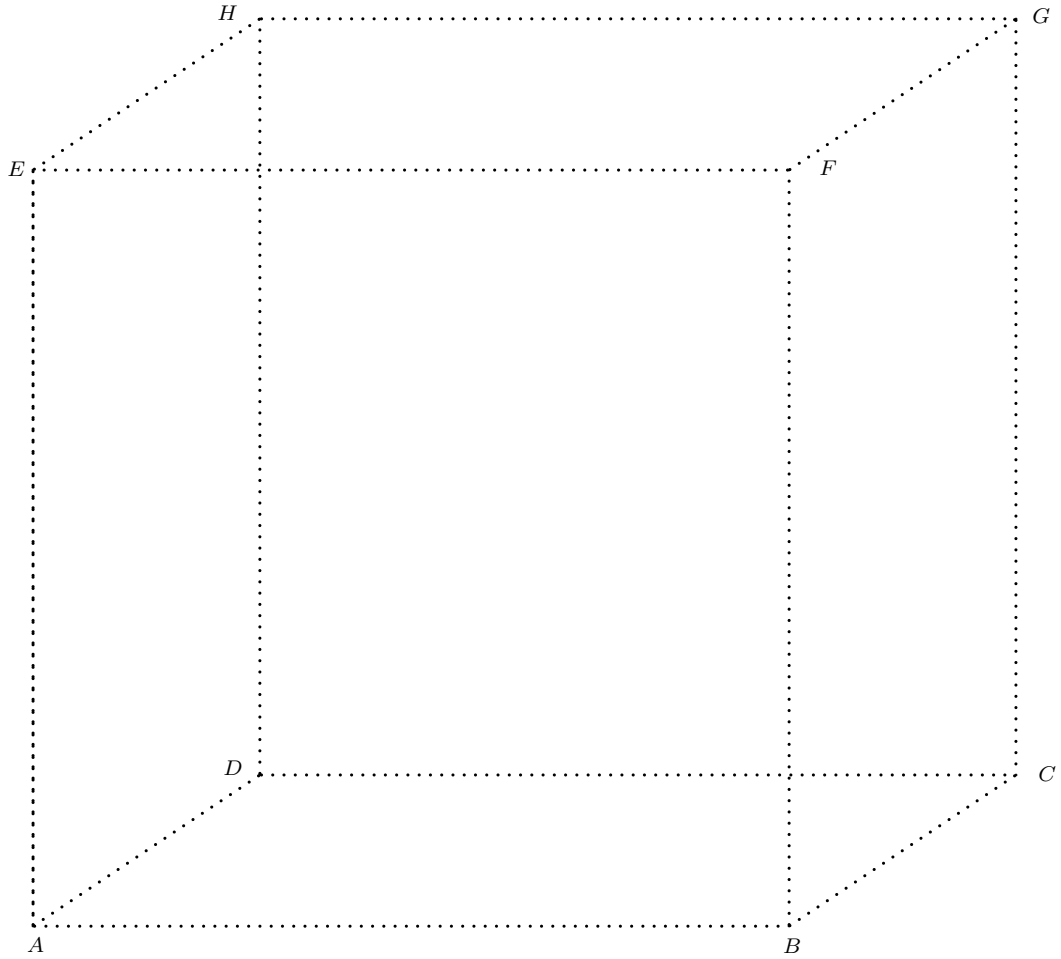
1.1.10



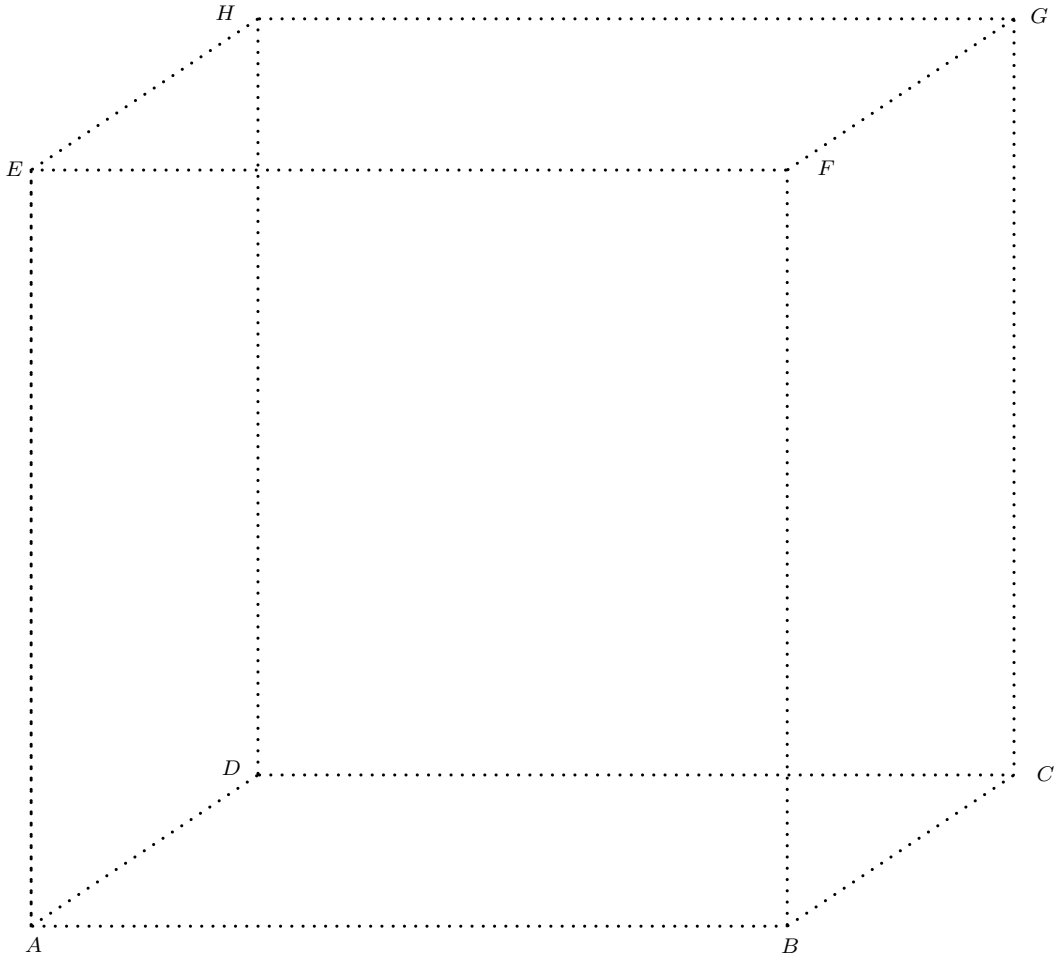
1.1.11



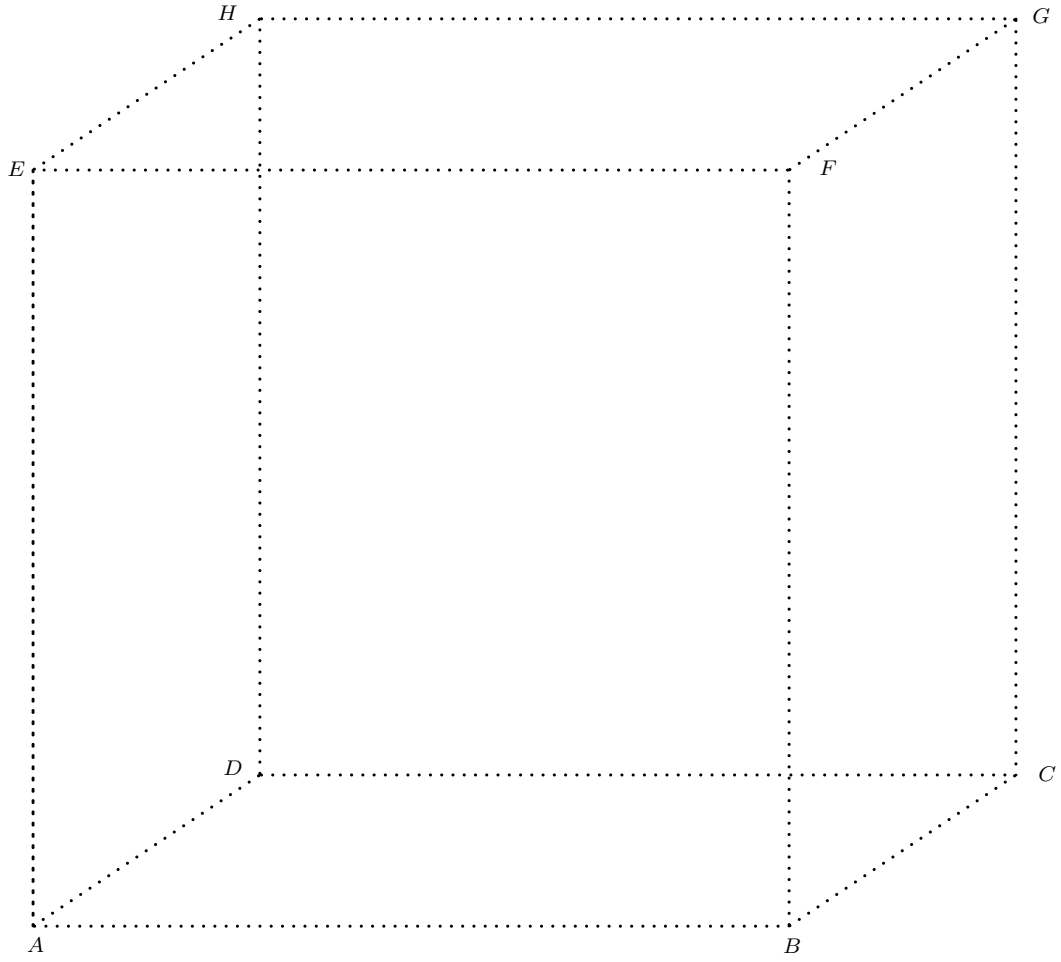
1.1.12



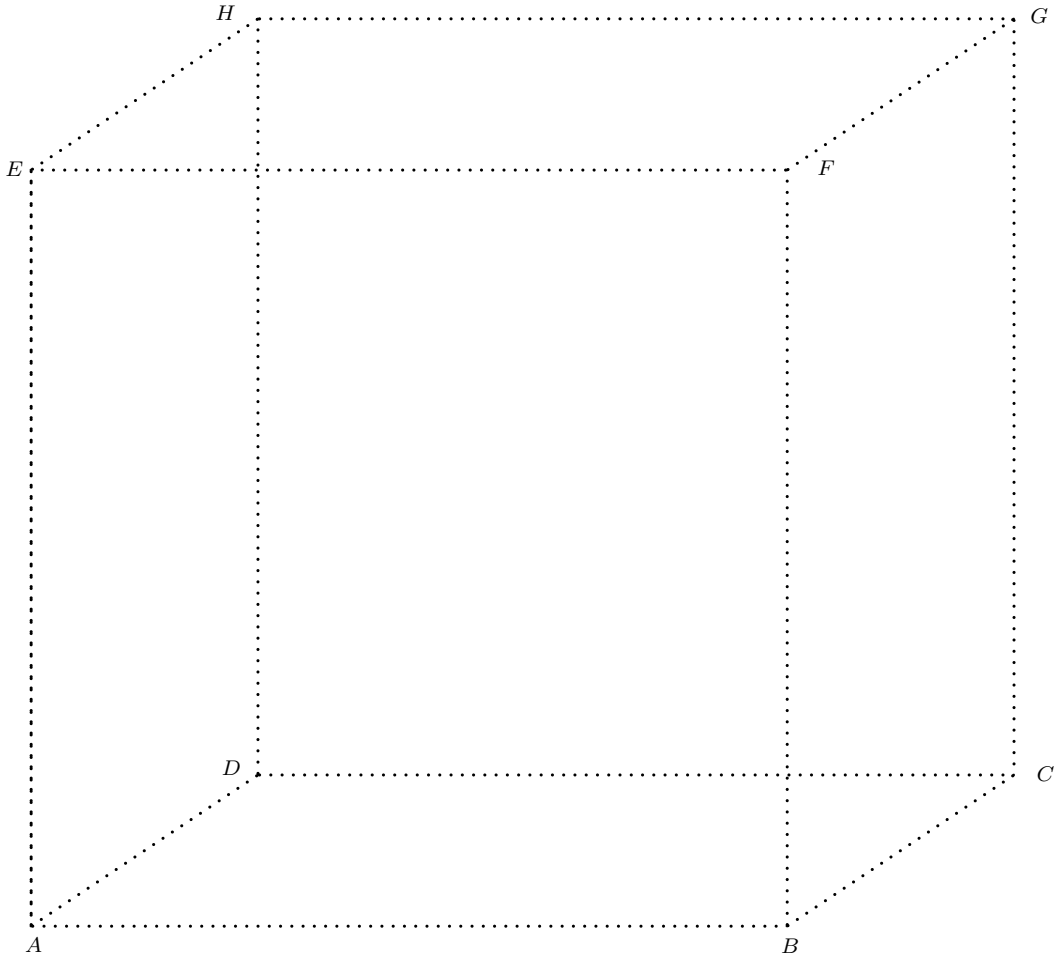
1.1.13



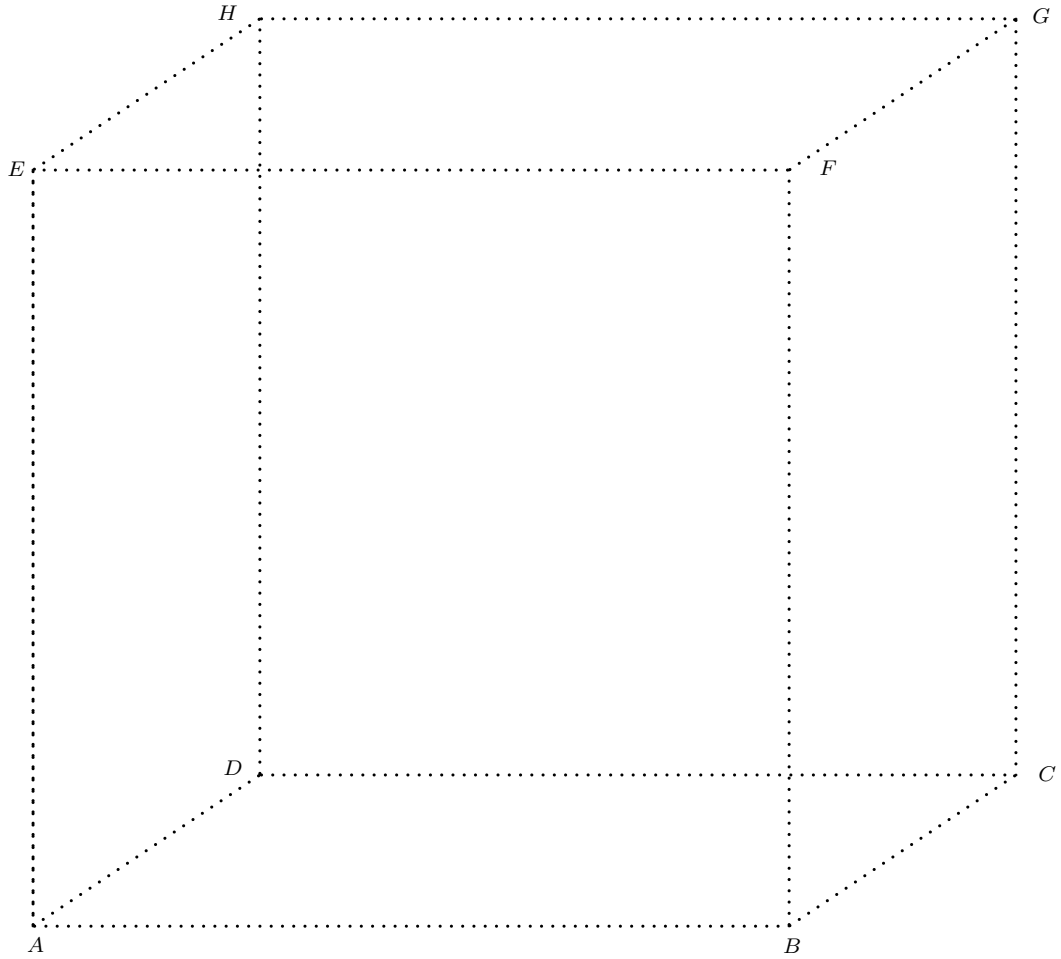
1.1.14



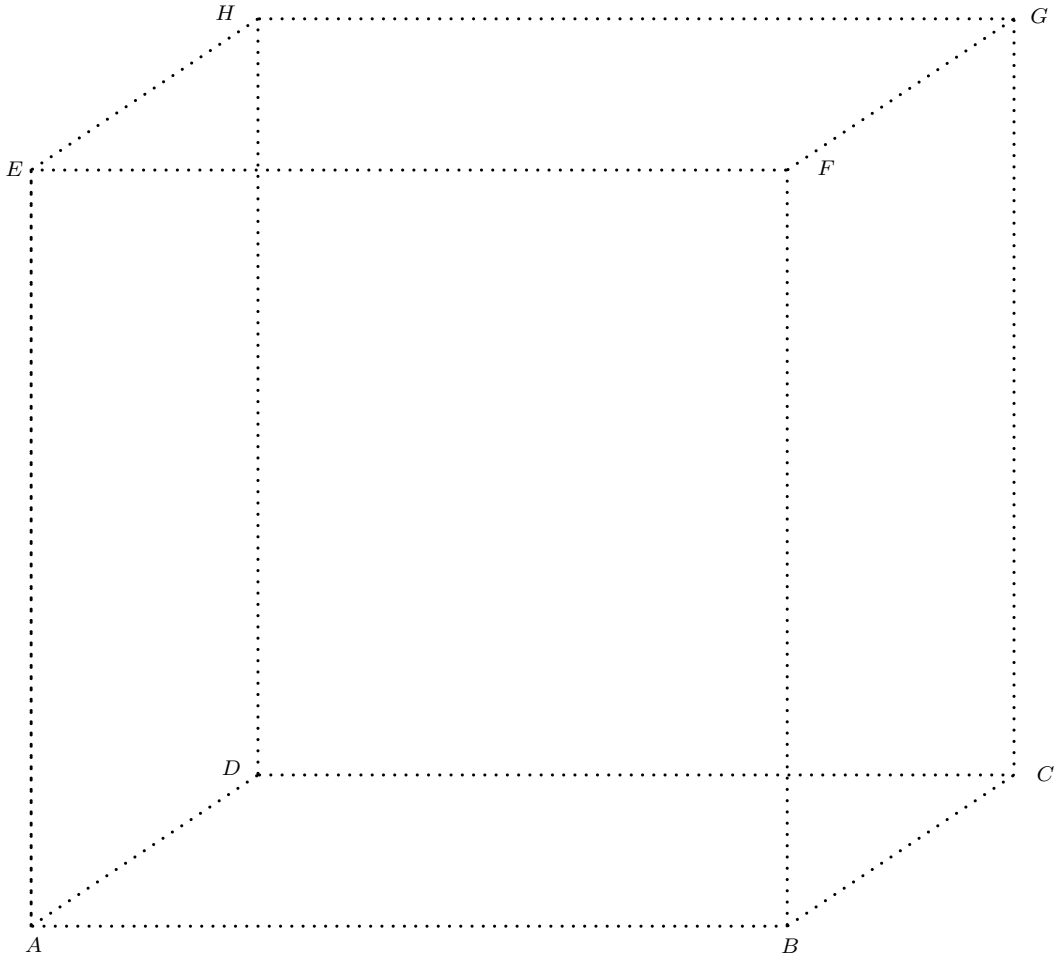
1.1.15



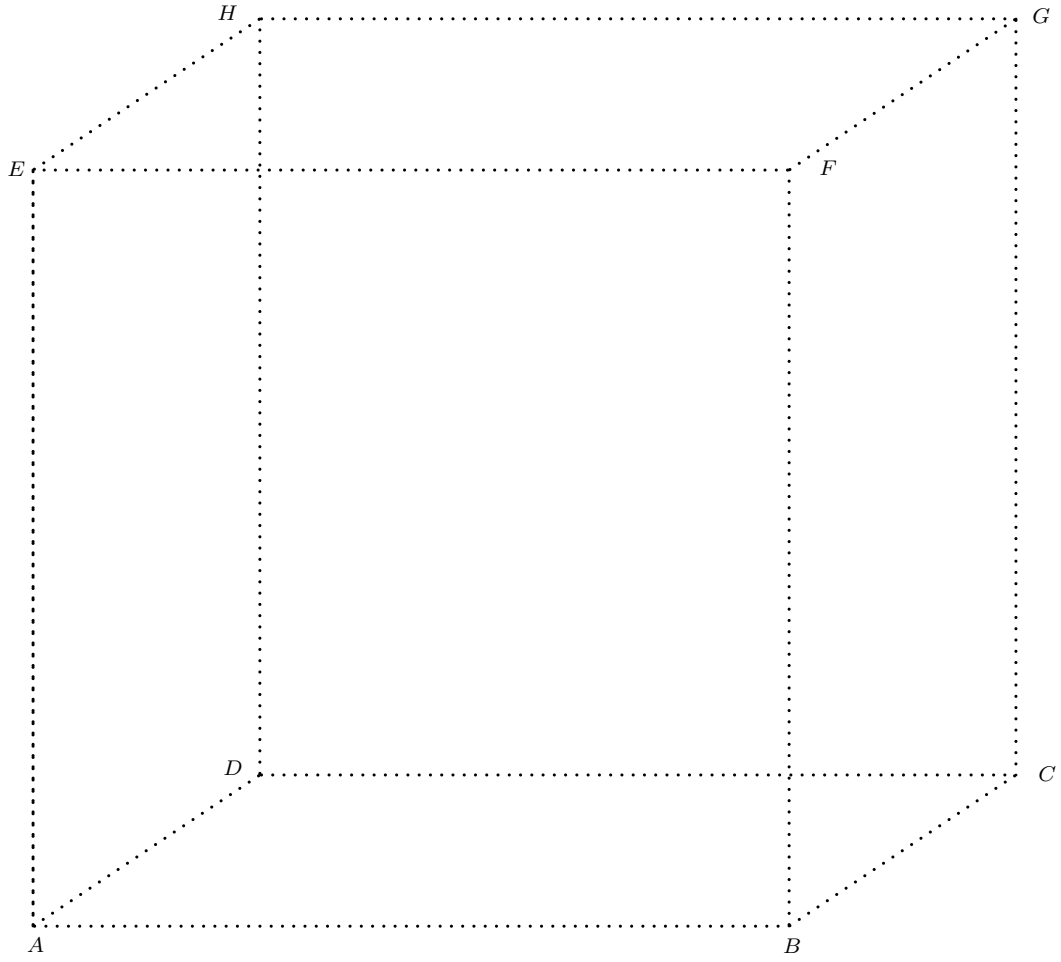
1.1.16



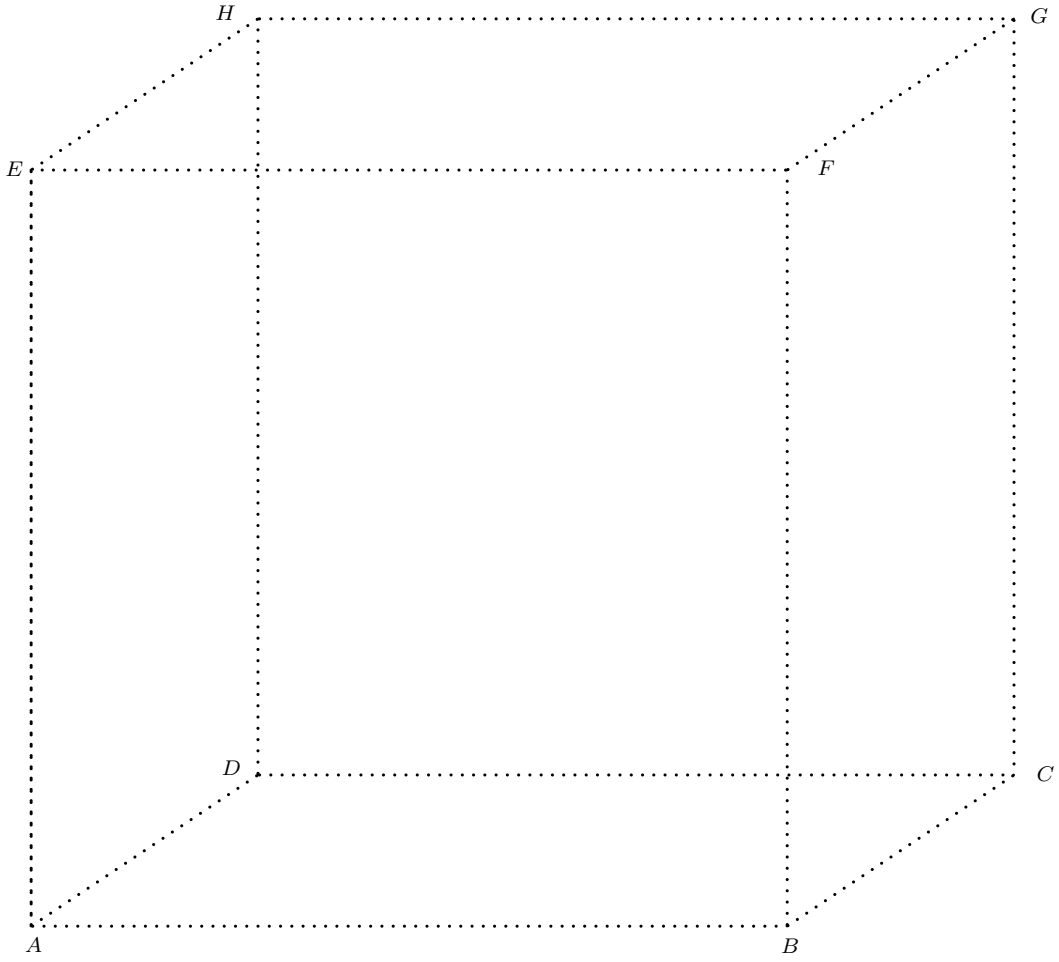
1.1.17



1.1.18



1.1.19



1.1.20 On a représenté ci-dessous le cube $ABCDEFGH$ dont le côté mesure 8 cm. Sur ce dessin, placer les milieux M, N, O et P des segments AB, AD, EF et EH, respectivement.

a) Dessiner le prisme $AMNEOP$ avec visibilité.

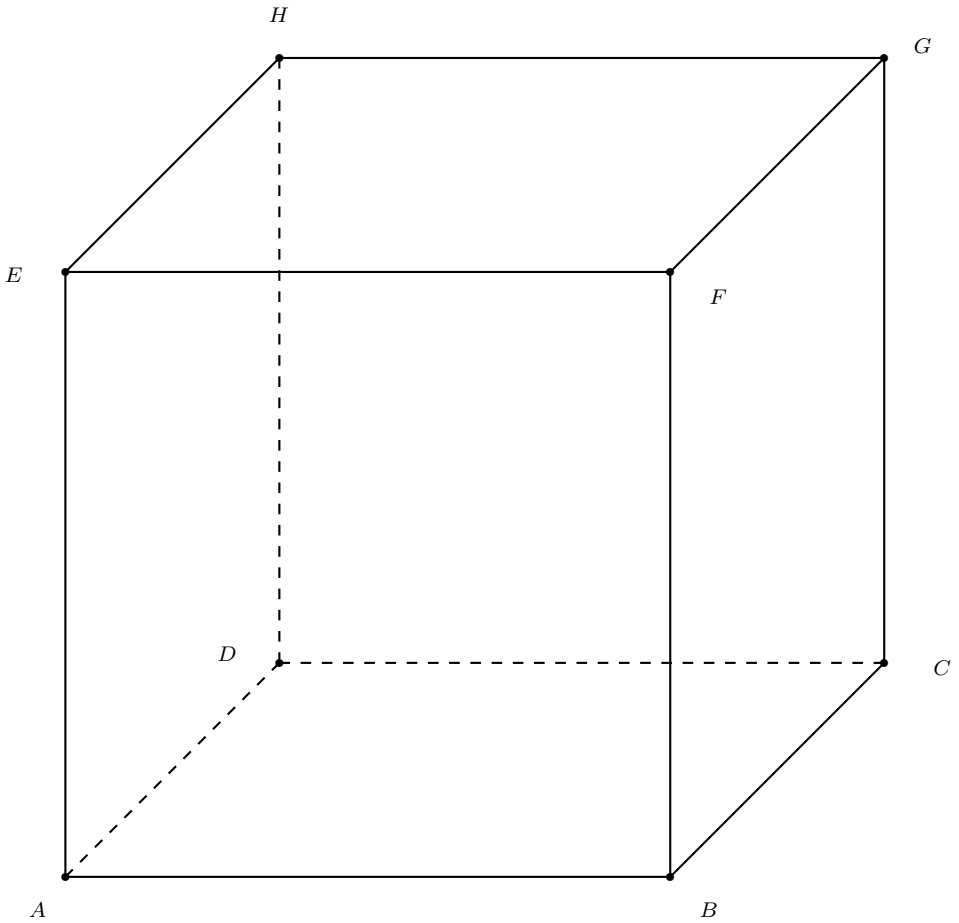
b) Calculer la longueur totale de ses arêtes.

c) Calculer son aire totale et son volume.

Placer encore les points Q, R, S et T, qui sont les milieux des segments BC, DC, FG et HG, respectivement.

a) Dessiner le prisme $BQRDFSTH$ avec visibilité.

b) Calculer son volume.



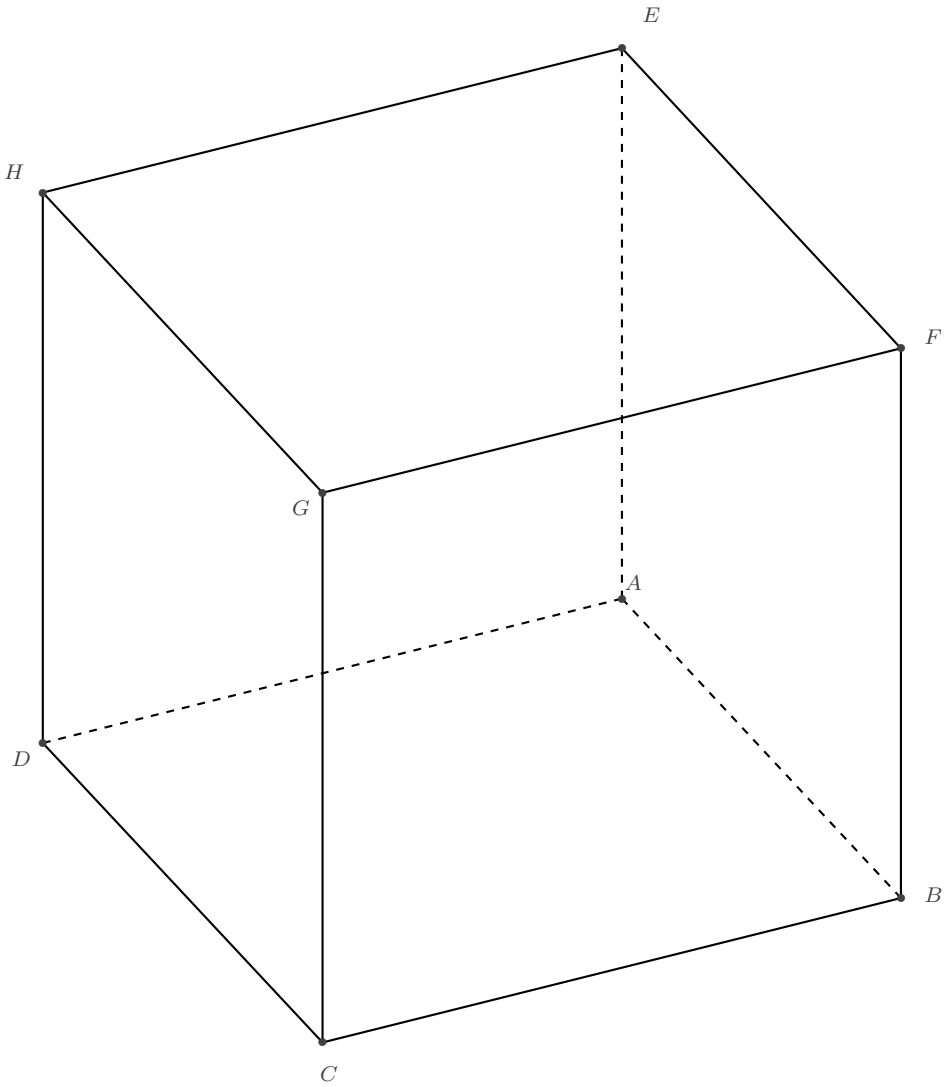
1.1.21 On a tracé ci-dessous à l'aide d'une axonométrie orthogonale, le cube $ABCDEFGH$ dont le côté mesure 10 cm.

a) Placer sur le dessin le milieu M de la face $EFGH$.

b) Dessiner la pyramide de sommet M et de base $ABCD$, avec visibilité.

c) Soit N le milieu de l'arête GC et O le milieu du segment GN. On considère le plan horizontal qui passe par O. Représenter les intersections de ce plan avec le cube.

d) Le plan en question découpe la pyramide $MABCD$ en deux polyèdres. Décrire ces polyèdres et calculer le volume et l'aire totale de chacun.

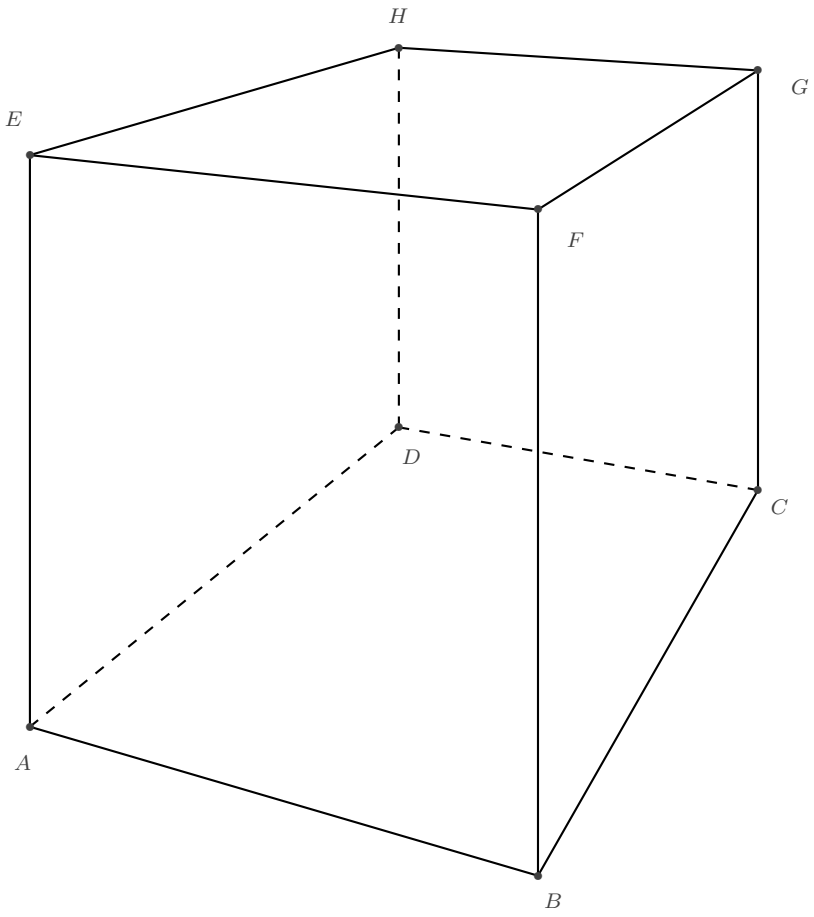


1.1.22 Dans le cube $ABCDEFGH$, représenté ci-dessous en perspective centrale, on considère le solide de sommets B, D, E et G.

a) Représenter ce solide.

b) De quel type de solide s'agit-il ? En donner une description précise.

c) Dessiner un développement de ce solide.



Chapitre 2

Introduction

2.1 Calculs

2.1.1 Calculer à l'aide de Python les éléments suivants. Afficher les résultats dans la console.

a) L'aire et le périmètre d'un rectangle dont les cotés mesurent 12.698 et 18.62.

b) Le volume d'une sphère de 15 cm de rayon.

c) Le nombre de secondes dans une année.

d) 2^{100}.

e) $\frac{\pi}{4} + \sqrt{5}$.

2.1.2 Calculer le volume d'un parallélépipède rectangle dont sont fournis au départ la largeur, la hauteur et la profondeur. Afficher le résultat dans la console.

```
largeur = 10
hauteur = 5.5
profondeur = 8.75
```

Modifier ensuite le programme pour que les variables soient entrées par l'utilisateur.

2.1.3 Calculer les solutions de l'équation $0.87\,x^2 - 6.02\,x - 2.005 = 0$.

2.1.4 Un gardien se rend 5 fois par jour au sommet du phare pour contrôler que tout est en ordre.
Il y a 327 marches pour arriver en haut du phare. On sait que la hauteur d'une marche est de 30.25 cm.
En une semaine, de combien de mètres est-il monté et descendu ? Afficher le résultat dans la console.

2.1.5 Écrire une fonction qui renvoie une valeur approchée de π en évaluant l'expression

$$\pi = 16 \cdot \arctan(1/5) - 4 \cdot \arctan(1/239)$$

dans laquelle arctan désigne la fonction arc tangente du module mathématique de python.

2.1.6 Écrire une fonction qui renvoie une valeur approchée de π en évaluant l'expression

$$\pi = \frac{\ln(640320^3 + 744)}{\sqrt{163}}$$

dans laquelle ln désigne la fonction `log` du module mathématique de python.

2.1.7 Dans le shell de python, évaluer l'expression

$$x^3 - x^2 + x - 1$$

pour $x = 0.457$.

2.1.8 Dans le shell de python, évaluer l'expression

$$1/x - x\,(1 - 1/x)$$

pour $x = 0.333$.

2.1.9 Soit $v = 1.4$. Vu que

$$1.4^2 = 1.96$$

on peut dire que v est une approximation de la racine de 2. On peut montrer que le nombre donné par

$$\frac{1}{2}\left(\frac{2}{v} + v\right)$$

est une « meilleure » approximation de la racine de 2. Dans le shell de python, calculer ce nombre.
En itérant ce processus dans le shell, trouver l'approximation du nombre $\sqrt{2}$ qui correspond le mieux à celle donnée par le shell python en évaluant l'expression

```
2**(1/2)
```

2.1.10 On considère la liste de nombres donnée par

$$1 + \frac{1}{2}, \qquad 1 + \frac{1}{2 + \frac{1}{2}}, \qquad 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2}}}, \qquad 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2}}}}, \quad \ldots$$

Calculer dans le shell python une valeur approchée de chacun de ces nombres.

2.1.11 Trouver la fraction continue qui donne le cinquième nombre de la liste de l'exercice précédent. Calculer une valeur approchée de ce nombre.

2.1.12 Dans le shell python, calculer la valeur de la racine de 2. Comparer avec la valeur trouvée à l'exercice précédent.

2.1.13 Écrire un programme en python qui permet de trouver le code décimal de la $n^{\text{ème}}$ valeur de la suite de l'exercice 2.1.10.

2.1.14 Quel terme de la suite de l'exercice 2.1.10 faut-il choisir pour obtenir un nombre dont toutes les décimales sont égales à celles du nombre affiché dans le shell de python pour la racine de 2 ?

2.2 Répétitions, conditions et fonctions

2.2.1 Initialiser deux entiers :

```
a = 0
b = 10
```

Écrire une boucle qui affiche a, puis qui ajoute 1 à la valeur de a tant qu'elle reste inférieure à celle de b.
Écrire une autre boucle qui soustrait 1 à la valeur de b et qui affiche sa valeur si elle est impaire. Boucler tant que b n'est pas nul.

2.2.2 Écrire des instructions qui calculent les trois sommes S_1, S_2 et S_3, pour $n = 100$. Afficher les résultats dans la console.

a) $S_1 = 1 + 2 + 3 + \cdots + n$, la somme des `n` premiers entiers.

b) $S_2 = 1^2 + 2^2 + 3^2 + \cdots + n^2$, la somme des `n` premiers carrés.

c) $S_3 = 1^3 + 2^3 + 3^3 + \cdots + n^3$, la somme des `n` premiers cubes.

2.2.3 Écrire des instructions qui renvoient la liste des carrés compris entre deux entiers `m` et `n`.

```
m = 51
n = 2022
```

Modifier ensuite le programme pour que les deux entiers `m` et `n` soient entrés par l'utilisateur.

2.2.4 Écrire des instructions qui affichent en colonne les 20 premiers termes de la table de multiplication par 13, en signalant au passage à l'aide d'un décalage vers la droite ceux qui sont également des multiples de 3.

2.2.5 Écrire des instructions qui affichent en colonne les n premiers termes de la table de multiplication par k, en signalant au passage à l'aide d'un décalage vers la droite ceux qui sont également des multiples de m. Les nombres n, m et k sont des entiers.

2.2.6 Écrire des instructions qui calculent les 50 premiers termes de la table de multiplication par 17, mais n'affiche que ceux qui sont des multiples de 7.

2.2.7 Écrire des instructions qui affichent une suite de 12 nombres tel que chaque terme soit égal au triple du terme précédent.

2.2.8 L'utilisateur donne un entier supérieur à 1 et le programme affiche, s'il y en a, tous ses diviseurs propres sans répétition ainsi que leur nombre. S'il n'y en a pas, il indique qu'il est premier. Par exemple :

```
>>> Entrez un entier strictement positif : 12
>>> Diviseurs propres sans répétition de 12 : 2 3 4 6 (soit 4 diviseurs propres)
>>> Entrez un entier strictement positif : 13
>>> Diviseurs propres sans répétition de 13 : aucun ! Il est premier.
```

2.2.9 Écrire un programme qui calcule la somme des chiffres d'un entier naturel entré par l'utilisateur. Afficher le résultat à la console.

2.2.10 Un nombre entier positif est **parfait** s'il est égal à la somme de tous ses diviseurs sauf lui-même. Par exemple 6 est un nombre parfait.
Écrire un programme qui recherche tous les nombres parfaits inférieurs à un nombre donné.

2.2.11 Deux nombres sont **aimables** si le premier est égal à la somme des diviseurs du deuxième sans celui-ci, et vice versa.
Écrire un programme qui recherche les nombres aimables.

2.2.12 Écrire une fonction qui retourne les solutions de l'équation du deuxième degré.

2.2.13 Soit la fonction réelle définie par $f(x) = -35 + 0.5\,x + 6.5\,x^2 + x^3$.

a) Écrire une fonction *maFonction* qui retourne $f(x)$ pour $x \in \mathbb{R}$.

b) Soit $a \in \mathbb{Z}$. Afficher à la console $f(a)$, pour $-10 \leqslant a \leqslant 10$.

c) Trouver le plus grand $b \in \mathbb{N}$ tel que $f(b) < 10000$.

2.2.14 La suite (u_n) est définie par $u_n = 2\sqrt{n} + 2$, pour tout $n \in \mathbb{N}$.

a) Écrire une fonction qui calcule u_n pour un n donné.

b) Afficher à la console les termes u_0 à u_{100}.

2.2.15 La suite (u_n) est définie par $u_n = \dfrac{5\,n^2 - 3}{n^2 + 1}$, pour tout $n \in \mathbb{N}$.

a) Écrire une fonction qui calcule u_n pour un n donné.

b) Afficher à la console les termes u_{10}, u_{100} et u_{1000}.

c) Qu'observe-t-on pour des valeurs de plus en plus grandes de n ?

2.2.16 Soit la suite définie par $u_0 = 10500$ et pour tout $n \in \mathbb{N}^\star$, $u_{n+1} = 0.99\,u_n + 150$.

a) Quelle est la plus petite valeur de n à partir de laquelle $u_n > 12000$.

b) Quelle est la plus petite valeur de n à partir de laquelle $u_n > 15000$.

2.2.17 Écrire une fonction qui prend trois entiers en paramètre et qui renvoie le plus grand des trois.

2.2.18 Faire calculer la somme des 100 premiers nombre pairs à l'aide d'une boucle.

2.2.19 Faire calculer la somme des 50 premiers nombres impairs.

2.2.20 Faire calculer la moyenne des 100 premiers nombres impairs.

2.2.21 Écrire une fonction qui renvoie la moyenne des n premiers entiers, n étant passé en paramètre.

2.2.22 Écrire une fonction `ma_factorielle` qui calcule le produit des n premiers nombres entiers, n étant passé en paramètre.
Calculer ensuite 10! à l'aide de cette fonction.

2.2.23 Chaque nombre de la suite de Fibonacci est la somme des deux nombres précédents. Les deux premiers nombres de cette suite sont 1 et 1. Calculer le dixième nombre de Fibonacci.

2.2.24 Ecrire une fonction `fibonacci` prenant un paramètre entier $n \geq 3$ et qui renvoie le $n^{\text{ème}}$ nombre de Fibonacci.

2.3 Polynômes

2.3.1 On considère la liste des coefficients du polynôme

$$p = x^5 - x^4 + x^3 - 3\,x^2 + 2\,x - 1$$

que l'on peut écrire, en python, sous la forme suivante :

```
p = [-1, 2, -3, 1, -1, 1]
```

a) Quels sont les polynômes qui correspondent aux listes suivantes ?
   - `a = [0,1,2,3]`
   - `b = [1,2,3]`
   - `c = [3,2,1]`
   - `d = [3,2,1,0,-1]`

b) Quelles sont les listes qui correspondent aux polynômes suivants ?
   - $e = x^2 + 2$
   - $f = 1 + 4\,x - x^3$
   - $g = x^5 - 1$

c) À l'aide d'une boucle `for`, faire afficher les éléments de `p` à la console, comme ci-dessous.

```
-1
2
-3
1
-1
1
```

d) À l'aide d'une boucle `while`, faire afficher à la console les éléments qui correspondent aux coefficients des puissances paires de x dans p, comme ci-dessous.

```
-1
-3
-1
```

2.3.2 Soit

$$p(x) = 2\,x^3 - 3\,x^2 + 5\,x - 1 \quad \text{et} \quad q(x) = 3\,x^3 + 2\,x^2 - 4\,x + 2$$

On représente ces deux objets mathématiques en python par les listes suivantes :

```
p = [-1, 5, -3, 2]
q = [2, -4, 2, 3]
```

a) Créer une liste `s` dans le shell python permettant de stocker les coefficients du polynôme $p + q$.

b) Pour calculer le premier terme de `s`, correspondant au terme constant de la somme, on écrit

```
s[0] = p[0] + q[0]
```

À l'aide d'instructions analogues, faire calculer les autres coefficients de la somme $p + q$.

c) Vérifier que la liste obtenue correspond bien au polynôme $p + q$.

2.3.3 Soit

$$p(x) = x^7 - x^6 + 3\,x^5 - 4\,x^4 - x^3 - 2\,x^2 - 3\,x - 1 \quad \text{et} \quad q(x) = 3\,x^3 + 2\,x^2 - 4\,x + 2$$

a) Donner les deux listes python `p` et `q`, correspondant à ces deux polynômes.

b) Créer une liste `s` dans le shell python permettant de stocker les coefficients du polynôme $p + q$. Cette liste aura la même longueur que celle qui correspond à p, le polynôme de plus haut degré des deux.
Pour calculer la somme, on écrit

```
s[i] = p[i] + q[i]
```

pour `i` inférieur ou égal au degré de q, puis on simplifie l'expression pour les termes restants. On écrit alors

```
s[i] = p[i]
```

pour `i` strictement supérieur au degré de q.

2.3.4 Soit $p(x) = x^2 + x + 2$ et $q(x) = x^3 - 2\,x$. Ecrire un programme python qui permet de calculer automatiquement

$$p + q, \quad p - q, \quad \text{et} \quad q - p$$

2.3.5 Soit

$$a(x) = -2\,x^2 + 2\,x - 1 \quad \text{et} \quad b(x) = 2\,x^3 - x^2 - 2\,x + 2$$

a) Dessiner le tableau qui illustre la multiplication de a par b.

b) Donner les deux listes python `a` et `b`, correspondant à ces deux polynômes.

c) Créer une liste `p` dans le shell python permettant de stocker les coefficients du polynôme $a \cdot b$. Quelle doit-être sa longueur ?

d) Pour calculer les quatre premiers éléments de `p`, on écrit

```
p[0] = a[0]*b[0]
p[1] = a[0]*b[1] + a[1]*b[0]
p[2] = a[0]*b[2] + a[1]*b[1] + a[2]*b[0]
p[3] = a[0]*b[3] + a[1]*b[2] + a[2]*b[1] + a[3]*b[0]
```

À l'aide d'instructions analogues, faire calculer les deux derniers coefficients du produit.

e) Vérifier que la liste obtenue correspond bien au polynôme $a \cdot b$.

2.3.6 Soit les polynômes

$$a(x) = 3\,x^2 - 4\,x + 3\ ,\ p(x) = x^4 + 2\,x^3 - 2\,x^2 - 4\,x + 17 \text{ et } q(x) = 2\,x^3 - 3\,x^2 - 5\,x + 18$$

Écrire un programme en python qui permet de

a) calculer $p - q$;

b) déterminer le degré du polynôme $p \cdot q$;

c) calculer et réduire au maximum a^2 et $a \cdot p \cdot q$;

d) déterminer le coefficient du polynôme $p \cdot q$ de degré 7 ;

e) déterminer le coefficient du polynôme $p \cdot q$ de degré 4 ;

f) déterminer les coefficients du polynôme $p \cdot q$.

2.3.7 Écrire une fonction `degre(p)` prend en paramètre la liste des coefficients d'un polynôme et retourne le degré de ce polynôme.

2.3.8 Écrire une fonction `affiche(p)` prend en paramètre la liste des coefficients d'un polynôme et qui affiche le polynôme dans la console.
Par exemple :

```
>>> affiche([4,0,3,-2])
>>> 4 + 0 X + 3 X^2 + -2 X^3
```

2.3.9 Écrire une fonction `somme(p, q)` prend en paramètre deux listes des coefficients correspondant à deux polynômes et qui retourne la liste des coefficients du polynôme qui est la somme des deux.

2.3.10 Écrire une fonction `evalue(p, x)` prend en paramètre la liste des coefficients correspondant au polynôme `p` et une valeur réelle `x` et qui retourne l'évaluation de `p` en `x`, c'est-à-dire $p(x)$.

2.3.11 Écrire une fonction `produit(p, q)` prend en paramètre la liste des coefficients correspondant au polynôme `p` et la liste des coefficients correspondant au polynôme `q` et qui retourne la liste des coefficients du produit des deux polynômes, c'est-à-dire $p \cdot q$.

2.4 Project Euler

Lien : https://projecteuler.net

2.4.1 (https ://projecteuler.net/problem=1)
If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.
Find the sum of all the multiples of 3 or 5 below 1000.

2.4.2 (https ://projecteuler.net/problem=3)
The prime factors of 13195 are 5, 7, 13 and 29.
What is the largest prime factor of the number 60085147514 ?

2.4.3 (https ://projecteuler.net/problem=5)
2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder.
What is the smallest positive number that is evenly divisible by all of the numbers from 1 to 20 ?

2.4.4 (https ://projecteuler.net/problem=6)
The sum of the squares of the first ten natural numbers is,

$$1^2 + 2^2 + \cdots + 10^2 = 385$$

The square of the sum of the first ten natural numbers is,

$$(1 + 2 + \cdots + 10)^2 = 55^2 = 3025$$

Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is

$$3025 - 385 = 2640$$

Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.

2.4.5 (https ://projecteuler.net/problem=9)
A Pythagorean triplet is a set of three natural numbers, $a < b < c$, for which,

$$a^2 + b^2 = c^2$$

For example, $3^2 + 4^2 = 9 + 16 = 25 = 5^2$.
There exists exactly one Pythagorean triplet for which $a + b + c = 1000$.
Find the product $a\,b\,c$.

2.4.6 (https ://projecteuler.net/problem=16)
$2^{15} = 32768$ and the sum of its digits is $3 + 2 + 7 + 6 + 8 = 26$.
What is the sum of the digits of the number 2^{1000} ?

2.4.7 (https ://projecteuler.net/problem=92)
A number chain is created by continuously adding the square of the digits in a number to form a new number until it has been seen before.
For example,
$44 \to 32 \to 13 \to 10 \to 1 \to 1$
$85 \to 89 \to 145 \to 42 \to 20 \to 4 \to 16 \to 37 \to 58 \to 89$
Therefore any chain that arrives at 1 or 89 will become stuck in an endless loop. What is most amazing is that EVERY starting number will eventually arrive at 1 or 89.
How many starting numbers below ten million will arrive at 89 ?

2.4.8 (https ://projecteuler.net/problem=206)
Find the unique positive integer whose square has the form `1_2_3_4_5_6_7_8_9_0`, where each `"_"` is a single digit.

2.5 Solutions des exercices

2.1.1

```
a = 12.698
b = 18.62
aire = a*b
perimetre = 2*(a+b)
print("aire = ", aire)
print("perimetre = ", perimetre)
# Exercice 1.1.1 b)
from math import pi
rayon = 15
volume = 4/3*pi*rayon**3
print("volume = ", volume)
# Exercice 1.1.1 c)
secondes = 365*24*60*60
print("nombre de secondes dans une annee = ", secondes)
# Exercice 1.1.1 d)
p = 2**100
print(p)
# Exercice 1.1.1 e)
from math import sqrt
q = pi/4 + sqrt(5)
print(q)
```

a) aire = 236.43676000000002

b) perimetre = 62.636

c) volume = 14137.166941154068

d) nombre de secondes dans une annee = 31536000

e) 1267650600228229401496703205376

f) 3.02146614089723

2.1.2

```
largeur = 10
hauteur = 5.5
profondeur = 8.75
volume = largeur*hauteur*profondeur
print("volume = ", volume)
# avec l'utilisateur
print("Entrez les dimensions :")
largeur = float(input("largeur = "))
hauteur = float(input("hauteur = "))
profondeur = float(input("largeur = "))
volume = largeur*hauteur*profondeur
print("volume = ", volume)
```

2.1.3

```
from math import sqrt
a = 0.87
b = -6.02
c = -2.005
delta = b**2-4*a*c
# les solutions
x_1 = (-1*b-sqrt(delta))/(2*a)
x_2 = (-1*b+sqrt(delta))/(2*a)
print(x_1, x_2)
```

Les solutions : -0.318404966945401 et 7.237945196830458

2.1.4

Il a parcouru 6924.224999999999 mètres.

2.1.5

```
import math
pi = 16 * math.atan(1/5) - 4 * math.atan(1/239)
print(pi)
```

2.1.6

```
import math
pi = math.log(640320**3 + 744)/math.sqrt(163)
print(pi)
```

2.1.7

```
x = 0.457
```

```
x**3-x**2+x-1
```

```
-0.656405007
```

2.1.8

```
x = 0.333
```

```
1/x-x*(1-1/x)
```

```
3.6700030030030026
```

2.1.9

```
v = 1.4
```

```
2**(1/2)
```

```
1.4142135623730951
```

```
v = 1/2*(2/v+v)

v

1.4142857142857141

v = 1/2*(2/v+v)

v

1.4142135642135643

v = 1/2*(2/v+v)

v

1.414213562373095
```

2.1.10

1.5, 1.4, 1.4166666666666667, 1.4137931034482758

2.1.11

```
1+1/(2+1/(2+1/(2+1/(2+1/2)))) = 1.4142857142857144
```

2.1.12

$1.4142857142857144 \simeq 1.4142135623730951$

Les quatre premières décimales sont identiques.

2.1.13

```
def rac2(n):
    i = 0
    x = .5
    while i < n:
        x = x + 2
        x = 1/x
        i = i + 1
    return 1 + x
```

2.1.14

```
from math import sqrt
r = sqrt(2)
def rac2(n):
    i = 0
    x = .5
    while i < n:
        x = x + 2
        x = 1/x
        i = i + 1
    return 1 + x
print(rac2(19))
print(rac2(20))
print(r)
```

2.2.1

```
a = 0
b = 10
while a < b:
    print("a = ",a)
    a = a + 1
while b > 0:
    b = b - 1
    if b%2 == 1:
        print("b = ",b)
```

2.2.2

```
n = 100
s_1, s_2, s_3 = 0, 0, 0
for k in range(101):
    s_1 = s_1 + k
    s_2 = s_2 + k**2
    s_3 = s_3 + k**3
print("S1 = ", s_1)
print("S2 = ", s_2)
print("S3 = ", s_3)
```

a) $S_1 = 5050$ b) $S_2 = 338350$ c) $S_3 = 25502500$

2.2.3

```
from math import sqrt
m = 51
n = 2022
a = int(sqrt(m)) + 1
b = int(sqrt(n))
for k in range(a, b + 1):
    print(k, "^2 = ", k**2)
# avec l'interaction
m = int("Entrez m = ")
n = int("Entrez n = ")
# idem ci-dessus
```

2.2.4

```
decalage = "      "
for j in range(1, 21):
    result = j * 13
    if result%3 == 0:
        print(decalage, j, " * 13 =", result)
    else:
        print(j, " * 13 =", result)
```

2.2.5

```
decalage = "      "
n = 20
k = 13
m = 3
for j in range(1, n + 1):
```

```
    result = j * k
    if result%m == 0:
        print(decalage, j, " * ", k, " = ", result)
    else:
        print(j, " * ", k, " = ", result)
```

2.2.6

```
for k in range(1, 51):
    result = k * 17
    if result%7 == 0:
        print(k, " * 17  = ", result, sep="")
```

2.2.7

```
debut = 17.2
for k in range(12):
    print(debut)
    debut = 3 * debut
```

2.2.8

```
from math import sqrt
entier = int(input("Entrez un entier strictement positif : "))
nombre_de_diviseur = 0
affichage = ""
for k in range(2, entier//2 + 1):
    if entier%k == 0:
        nombre_de_diviseur = nombre_de_diviseur + 1
        affichage = affichage + " " + str(k)
if nombre_de_diviseur > 0:
    affichage = affichage + " (soit " + str(nombre_de_diviseur)
                                              + " diviseurs propres)"
else:
    affichage = "aucun ! Il est premier."

print("Diviseurs propres sans repetition de ", entier, " : ",
                                                 affichage, sep = "")
```

2.2.9

```
entier = int(input("Entrez in nombre : "))
chaine = str(entier)
somme = 0
for caratere in chaine:
    somme = somme + int(caratere)
print(somme)
```

2.2.10

```
borne = 500
for nombre in range(2, borne):
    somme = 0
    for i in range(1, nombre//2+1):
        if nombre%i == 0:
            somme = somme + i
    if nombre == somme:
        print(nombre)
```

2.2.11

```
def somme_diviseurs(nombre):
    somme = 0
    for k in range(1, nombre):
        if nombre%k == 0:
            somme = somme + k
    return somme
def nombres_amicaux(p, q):
    a = somme_diviseurs(p)
    b = somme_diviseurs(q)
    if a == q and b == p:
        return True
    else:
        return False
def amicaux(borne):
    i, j = 2, 1
    while i < borne:
        j = 1
        while j < i:
            if nombres_amicaux(i, j):
                print(i, j)
            j = j + 1
        i = i + 1
amicaux(1220)
```

Ce code n'est pas optimisé !

2.2.12

```
from math import sqrt
print("Entrez les cofficients a, b et c")
a = int(input("a = "))
b = int(input("b = "))
c = int(input("c = "))
delta = b**2 - 4*a*c
if delta < 0:
    print("L'equation n'a pas de solution")
elif delta == 0:
    x = -b/(2*a)
    print("Une solution : ", x, sep = "")
else:
    x_1 = (-b - sqrt(delta))/(2*a)
    x_2 = (-b + sqrt(delta))/(2*a)
    print("Deux solutions :", x_1, " et ", x_2, sep = "")
```

2.2.13

```
def maFonction(x):
    return -35 + 0.5*x+6.5*x**2+x**3
# Exercice 2.2.13 b)
for n in range(-10, 11):
    print(maFonction(n))
# Exercice 2.2.13 c)
b = 0
while maFonction(b) < 10000:
    b = b + 1
print(b)
```

2.2.14

```
from math import sqrt
def u(n):
    return 2*sqrt(n)+2
# Exercice 2.2.14 b)
for n in range(101):
    print(u(n))
```

2.2.15

```
def u(n):
    return (5*n**2 - 3)/(n**2 + 1)
# Exercice 2.2.15 b)
print(u(10), u(100), u(1000))
```

c) Les valeurs de $u(n)$ se rapprochent de 5.

2.2.16

```
def u(n):
    u = 10500
    for k in range(n + 1):
        u = 0.99*u + 150
    return u
# Exercice 2.2.16 a)
n = 1
while u(n) < 12000:
    n = n + 1
print(n)
```

a) $n = 40$

b) $\forall n \in \mathbb{N}$, $u(n) < 15000$.

2.2.17

```
def max(x, y, z):
    if x < y:
        x, y = y, x
    if x < z:
        x, z = z, x
    return x
print(max(12,40,-15))
def Max(a,b):
    if a > b:
        return a
    else:
        return b
def Maxi(a, b, c):
    return Max(Max(a,b), c)
print(Maxi(120,40,150))
```

2.2.18

```
acc = 0
terme = 0
while terme < 101:
    acc += 2*terme
    terme += 1
print('la somme des 100 premiers nombre pairs : ', acc)
```

2.2.19

```
acc = 0
compteur = 1
while compteur < 51:
    acc += (2*compteur - 1)
    compteur += 1
print(' la somme des 50 premiers nombres impairs', acc)
```

2.2.20

```
acc = 0
compteur = 1
while compteur < 101:
    acc += (2*compteur - 1)
    compteur += 1
print('la somme des 100 premiers nombres impairs', acc)
print('la moyenne des 100 premiers nombres impairs', acc/100)
```

2.2.21

```
def moyenne(n):
    acc = 0
    terme = 0
    while terme <= n:
        acc += terme
        terme += 1
    return acc/n
print(moyenne(10))
```

2.2.22

```
N = 10
def ma_factorielle(n):
    fac = 1
    if n == 0:
        return fac
    else:
        while n > 0:
            fac *= n
            n -= 1
        return fac
print(ma_factorielle(N))

def ma_factorielle_recursive(n):
    if n == 0:
        return 1
    else:
        return n * ma_factorielle_recursive(n-1)
print(ma_factorielle_recursive(N))
```

2.2.23

```
def fibonacci(f1 = 1, f2 = 1, n = 9):
    while n > 0:
        f1, f2 = f2, f1 + f2
        n -= 1
    return f1
print(fibonacci(1,1,19))

# Example 1: Using looping technique
def fib(n):
 a,b = 1,1
 for i in range(n-1):
  a,b = b,a+b
 return a
print(fib(10))

# Example 2: Using recursion
def fibR(n):
 if n==1 or n==2:
  return 1
 return fibR(n-1)+fibR(n-2)
print(fibR(10))

# avec une liste
liste = []
for k in range(100):
    liste.append(fibonacci(1, 1, k))

for k in range(2, 100):
    print(fibonacci(1,1,k), fibonacci(1,1,k-1), fibonacci(1,1,k)/fibonacci(1,1,k-1)
```

2.2.24

```
def fibonacci(f1 = 1, f2 = 1, n = 9):
    while n > 0:
        f1, f2 = f2, f1 + f2
        n -= 1
    return f1
```

2.3.1

a) $a = 3\,x^3 + 2\,x^2 + x$, $b = 3\,x^2 + 2\,x + 1$, $c = x^2 + 2\,x + 3$ et $d = -x^4 + x^2 + 2\,x + 3$

b) `e=[2,0,1]`, `f=[1,4,0,-1]` et `e=[-1,0,0,0,0,1]`

c)

```
p = [-1, 2, -3, 1, -1, 1]
print("Tous les coefficients")
for c in p:
    print(c)
print("Les coefficients des puissances paires de x")
i = 0
deg_p = len(p) - 1

k = deg_p if deg_p%2 == 0 else deg_p - 1
while i <= k:
    print(p[i])
    i = i + 2
```

2.3.2

```
p = [-1, 5, -3, 2]
q = [2, -4, 2, 3]
##a)
s = [0 for i in range(max(len(p), len(q)))]
s
[0, 0, 0, 0]
##b)
s[0] = p[0] + q[0]
s[1] = p[1] + q[1]
s[2] = p[2] + q[2]
s[3] = p[3] + q[3]
s
[1, 1, -1, 5]
```

On voit que la liste `s` correspond bien au polynôme $p(x) + q(x) = 5\,x^3 - x^2 + x + 1$.

2.3.3

```
p = [-1, -3, -2, -1, -4, 3, -1, 1]
q = [2, -4, 2, 3]
s = [0 for i in range(max(len(p), len(q)))]
s
[0, 0, 0, 0, 0, 0, 0, 0]
for i in range(len(q)):
    s[i] = p[i] + q[i]
for i in range(len(q), len(p)):
    s[i] = p[i]
s
[1, -7, 0, 2, -4, 3, -1, 1]
```

On voit que la liste `s` correspond bien au polynôme

$$p(x) + q(x) = x^7 - x^6 + 3\,x^5 - 4\,x^4 + 2\,x^3 - 7\,x + 1$$

2.3.4

```
p = [2, 1, 1]
q = [0, -2, 0, 1]
s = [0 for i in range(max(len(p), len(q)))]
for i in range(len(q)):
    s[i] = p[i] + q[i]
for i in range(len(q), len(p)):
    s[i] = q[i]
d = [0 for i in range(max(len(p), len(q)))]
for i in range(len(q)):
    d[i] = p[i] - q[i]
for i in range(len(q), len(p)):
    d[i] = -q[i]
```

2.3.5

```
a = [-1, 2, -2]
b = [2, -2, -1, 2]
deg = (len(a) - 1) + (len(b) - 1)
p = [0 for i in range(deg + 1)]
p[0] = a[0]*b[0]
p[1] = a[0]*b[1] + a[1]*b[0]
p[2] = a[0]*b[2] + a[1]*b[1] + a[2]*b[0]
p[3] = a[0]*b[3] + a[1]*b[2] + a[2]*b[1]
p[4] = a[1]*b[3] + a[2]*b[2]
p[5] = a[2]*b[3]
p_ = [0 for i in range(deg + 1)]
for k in range(len(p_)):
    for i in range(len(a)):
        for j in range(len(b)):
            if (i + j) == k:
                p_[k] += a[i]*b[j]
```

```
##Vérification p = p_= [-2, 6, -7, 0, 6, -4]

import sympy as sp

x = sp.var("x")

sp.expand((-2*x**2+2*x-1)*(2*x**3-x**2-2*x+2))

-4*x**5 + 6*x**4 - 7*x**2 + 6*x - 2
```

2.3.6

a) $p - q = x^4 + x^2 + x - 1$

b) de degré 7

c) $a^2 =$ `9 +  -24 X^1 +  34 X^2 +  -24 X^3 +  9 X^4`

$a \cdot p \cdot q =$ `918 +  -1695 X^1 +  1345 X^2 +  73 X^3 +`

`-551 X^4 +  207 X^5 + 81 X^6 +  -43 X^7 +  -5 X^8 +  6 X^9`

d) 2

e) 6

f) $306 + -157X^1 + -67X^2 + 92X^3 + 6X^4 + -15X^5 + 1X^6 + 2X^7$

2.3.7

```
def degre(p) :
    return len(p)-1
```

2.3.8

```
def affiche(p) :
    q = str(p[0])
    for j in range(1, degre(p)+1):
        q = q + " +  " + str(p[j]) + " X^" + str(j)
    print(q)
```

2.3.9

```
def somme(p,q):
    deg = max(degre(p),degre(q))
    add = [0] * (deg+1)
    for i in range(0, degre(p)+1):
        add[i] = p[i]
    for j in range(0, degre(q)+1):
        add[j] = add[j] + q[j]
    return add
```

2.3.10

```
def evalue(p,x):
    p_x = p[0]
    for k in range(1, len(p)):
        p_x = p_x + p[k]*x**k
    return p_x
```

2.3.11

```
def degre(p) :
    return len(p)-1

def somme(p,q):
    deg = max(degre(p),degre(q))
    add = [0] * (deg+1)
    for i in range(0, degre(p)+1):
        add[i] = p[i]
    for j in range(0, degre(q)+1):
        add[j] = add[j] + q[j]
    return add

def affiche(p) :
    q = str(p[0])
    for j in range(1, degre(p)+1):
        q = q + " +  " + str(p[j]) + " X^" + str(j)
    print(q)

def produit(p, q):
    deg = degre(p) + degre(q)
    mul = [0] * (deg+1)
    for i in range(degre(p)+1):
        for j in range(degre(q)+1):
            mul[i+j] = mul[i+j] + p[i] * q[j]
    return mul
```

2.4.1

```
>>> 233168
```

2.4.2

```
>>> 6857
```

2.4.3

```
>>> 232792560
```

2.4.4

```
>>> 25164150
```

2.4.5

```
>>> 31875000
```

2.4.6

```
>>> 1366
```

2.4.7

```
>>> 8581146
```

2.4.8

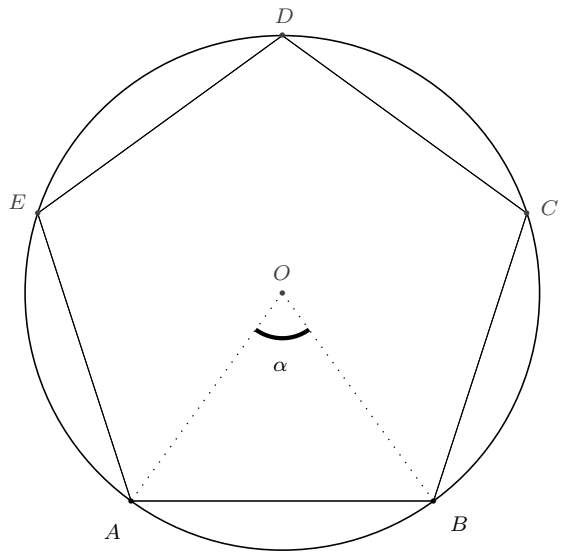
```
>>> 1389019170
```

Chapitre 3

Approcher π

3.1 L'approche d'Archimède

3.1.1 Soit P_n le polygone à n côtés inscrit dans un cercle de rayon r. Notons α l'angle au centre de l'arc déterminé par l'un des côtés de P_n.



Montrer que l'on a

$$\pi \simeq n \cdot \sin\left(\frac{\alpha}{2\,n}\right)$$

si n est assez grand.

3.1.2 En utilisant l'exercice précédent, les fonctions `math.sin` et `math.radians`, programmer une fonction `archi_pi` qui prend un nombre entier en paramètre et qui renvoie une valeur approchée de π.

3.1.3 Quelle valeur de `n` faut-il choisir pour que `archi_pi(n)` renvoie la même valeur que `math.pi` ?

3.2 Autres approches

3.2.1 Ecrire une fonction `leibniz` qui prend un paramètre entier n et qui renvoie une approximation de π basée sur la formule ci-dessous :

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \dots$$

3.2.2 Faire tourner la fonction `leibniz` avec 10 000 termes, puis 100 000 termes.

3.2.3 Faire afficher l'approximation de π donnée par `math.pi`. Peut-on obtenir en un temps raisonnable le même nombre de décimales avec notre fonction `leibniz` ?

3.2.4 Comparer le résultat donné par la fonction `leibniz` pour n termes avec celui donné par la fonction `archi_pi` pour n côtés.

3.2.5 Ecrire une fonction `wallis` qui prend un paramètre entier n et qui renvoie une approximation de π basée sur la formule ci-dessous :

$$\frac{\pi}{2} = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \dots$$

3.2.6 Faire tourner la fonction `wallis` avec 10 000 termes, puis 100 000 termes.

3.2.7 Faire afficher l'approximation de π donnée par `math.pi`. Peut-on obtenir en un temps raisonnable le même nombre de décimales avec notre fonction `wallis` ?

3.2.8 Comparer le résultat donné par la fonction `wallis` pour n termes avec celui donné par la fonction `archi_pi` pour n côtés et celui donné par la fonction `leibniz` pour n termes.

3.2.9 Écrire une fonction qui renvoie une valeur approchée de π en évaluant l'expression

$$\pi = \sqrt{12}\left(1 - \frac{1}{3 \cdot 3} + \frac{1}{5 \cdot 3^2} - \frac{1}{7 \cdot 3^3} + \dots\right)$$

Utiliser un paramètre nommé n pour désigner le nombre de termes à inclure dans la somme.

3.3 Une simulation de type « Monte Carlo »

3.3.1 Écrire une fonction `monteCarlo` qui prend en paramètre un nombre entier et qui renvoie une approximation de π. Cette fonction génère n couples de nombres réels appartenant à $[0;1] \times [0;1]$ et calcule l'approximation de π en comptant ceux qui se trouvent dans le quart de disque inclus dans le carré $[0;1] \times [0;1]$.

3.3.2 Écrire un programme qui fait se déplacer une tortue aléatoirement sur l'écran, dans un carré de 200 pixels de côté, dont le coin inférieur gauche est placé à l'origine. La tortue se déplace continuellement, tant que le programme n'a pas été interrompu par l'utilisateur.

3.3.3 Modifier le programme de l'exercice précédent de manière à ce que la tortue marque un point à l'écran à la fin de chaque déplacement. Le point sera de couleur bleue si l'emplacement est dans le disque de rayon 200 pixels centré à l'origine et de couleur rouge sinon.

3.3.4 Faire en sorte que le programme de l'exercice précédent garde un décompte du nombre total de points marqués par la tortue et également du nombre de points bleus. En déduire une approximation de π.

3.3.5 Écrire une fonction `monteCarlo` qui prend en paramètres une tortue, le côté d'un carré exprimé en pixels, un nombre de points à marquer au total, et qui renvoie une approximation de π.

3.3.6 Faire tourner la fonction `monteCarlo`, avec ou sans tortue, en augmentant le nombre de points. Tester les fonctions pour 10 000 points, 100 000 points et plus si possible.

3.3.7 Comparer l'approximation de π donnée par la fonction `monteCarlo` avec celles données par les fonctions `wallis` ou `leibniz`.

3.3.8 Écrire une fonction `estDansDisque` qui prend un point du plan et un rayon en paramètres. La fonction renvoie `True` si le point se trouve dans le disque et `False` sinon.

3.3.9 Modifier la fonction `monteCarlo` de sorte à ce qu'elle utilise la fonction auxiliaire `estDansDisque`.

3.4 Solutions des exercices

3.1.1 En classe.

3.1.2

```
from math import sin
from math import radians
def archi_pi(n):
    angle_interne = 360.0/n
    demi_angle = angle_interne/2
    demi_cote = sin(radians(demi_angle))
    cote = demi_cote * 2
    perimetre = n * cote
    pi = perimetre/2
    return pi
if __name__ == "__main__":
    print(archi_pi(50))
```

3.1.3

```
import math, Ex_3_1_2
pi = math.pi
archi_pi = Ex_3_1_2.archi_pi
for k in range(int(1e+7), int(1e+8), 50000):
    delta = pi - archi_pi(k)
    print(k, pi, archi_pi(k), delta)
```

3.2.1

```
import math
pi = math.pi

def leibniz(n):
    acc = 0
    terme = 1
    if n == 0:
        return 1
    else:
        while terme < n:
            acc = acc + (-1)**(terme+1)/(2*terme-1)
            terme += 1
        return 4*acc

# variante
def leibnitz_2(terms):
    acc = 0
    num = 4
    den = 1
    for aterm in range(terms):
        nextterm = num/den * (-1)**aterm
        acc += nextterm
        den += 2
    return acc

if __name__ == "__main__":
    print(leibniz(100))
    print(leibnitz_2(1000))
```

3.2.2

```
import Exo_3_2_8
print(Exo_2_3_8.leibniz(10000))
print(Exo_2_3_8.leibniz(100000))
```

3.2.3

```
import math, Exo_3_2_8
leibniz = Exo_3_2_8.leibniz
PI = math.pi
valeur = 1e4
while Exo_2_3_8.leibniz(valeur)-PI > 1e-8:
    print(leibniz(valeur)-PI)
    valeur = valeur *10
print(leibniz(valeur))
print(PI)
print(valeur)
```

Non !

3.2.4

```
import math, Exo_3_1_2, Exo_3_3_8
pi = math.pi
leibniz = Exo_3_2_8.leibniz
archi_pi = Exo_3_1_2.archi_pi
for k in range(8, 1000, 8):
    delta = archi_pi(k) - leibniz(k)
    print(k, pi, archi_pi(k), leibniz(k), delta)
```

3.2.5

```
# pi / 2 = (2/1*2/3) * (4/3 * 4/5) * (6/5*6/7) * (8/7*8/9) * ....
#           terme 1     terme 2       terme 3     terme 4
def wallis(nombre_de_termes):
    acc = 1
    num = 2
    for k in range(nombre_de_termes):
        terme_gauche = num/(num-1)
        terme_droite = num/(num+1)
        acc = acc * terme_gauche * terme_droite
        num = num + 2
    return 2 * acc
if __name__ == '__main__':
    print(wallis(500))
```

3.2.6

```
import Exo_3_2_12
wallis = Exo_3_2_12.wallis
# Faire tourner la fonction leibniz avec 10 000 termes
print(wallis(10000))
# Faire tourner la fonction leibniz avec 100 000 termes
print(wallis(100000)
```

3.2.7

```
import time, math, Exo_3_2_12
pi = math.pi
wallis = Exo_3_2_12.wallis
delta = 1e-9

nombre_de_termes = 1000
print('BEGIN')
temps_depart = time.time()
while pi - wallis(nombre_de_termes) > delta:
    nombre_de_termes = nombre_de_termes * 10
    print(wallis(nombre_de_termes))
temps_fin = time.time()
print('END')

print('pi = ', pi, '  ;  wallis(', nombre_de_termes, ') = ',
                                        wallis(nombre_de_termes), sep='')

temps = int(temps_fin-temps_depart)
minutes  = temps//60
secondes = temps%60
print('Duree : ', minutes, ' minutes ; ', secondes, ' secondes', sep = '')
```

3.2.8

```
import time, math, Exo_3_1_2, Exo_3_2_8, Exo_3_2_12
pi = math.pi
archi_pi = Exo_3_1_2.archi_pi
leibniz  = Exo_3_2_8.leibniz
wallis   = Exo_3_2_12.wallis
n  = 100000000
n2 = 160693804425899027554196209234116260252220299378279283530137 6
a = archi_pi(n2)
l = leibniz(n)
w = wallis(n)
print(a, l, w, sep=' ; ')
print(pi - a, pi - l, pi - w, sep=' ; ')
```

3.2.9

```
import math
def app_pi_le_retour(n):
    acc = 0
    num = 1
    for k in range(n):
        den = (-1)**(k) * (2*k+1) * 3**(k)
        acc = acc + num/den
        print(k, acc)
    return math.sqrt(12)*acc
```

3.3.1

```
import random, math
def monteCarlo(n):
    compteur  = 0
    for k in range(n):
        x = random.random()
        y = random.random()
        distance = math.sqrt(x**2 + y**2)
```

```
        if distance <= 1:
            compteur += 1
    return compteur/n
pi_appr = 4 * monteCarlo(2000000)
print(pi_appr)
```

3.3.2

```
import turtle, random
wn = turtle.Screen()
wn.bgcolor('gray')
wn.setworldcoordinates(0,0,2,2)
bob = turtle.Turtle()
bob.up()
bob.goto(1,1)
bob.down()
while 1:
    angle = random.randint(-180,180)
    bob.setheading(angle)
    bob.fd(0.2)
    if (bob.xcor() >= 2 or bob.xcor() <=0 or bob.ycor() >= 2 or bob.ycor() <= 0):
        bob.goto(1, 1)
```

3.3.3 Voir 3.3.6

3.3.4 Voir 3.3.6

3.3.5 Voir 3.3.6

3.3.6

```
import turtle, random

def axes(uneTortue, cote):
    uneTortue.up()
    uneTortue.goto(-cote, 0)
    uneTortue.down()
    uneTortue.forward(2*cote)
    uneTortue.up()
    uneTortue.goto(0, -cote)
    uneTortue.left(90)
    uneTortue.down()
    uneTortue.forward(2*cote)
    uneTortue.up()
    uneTortue.right(90)
    uneTortue.down()

def monte_carlo(uneTortue, cote, nombre_de_points):
    uneTortue.up()
    points_interieurs = 0
    j = 0
    while j < nombre_de_points:
        x = cote * random.random()
        y = cote * random.random()
        uneTortue.goto(x, y)
        distance_au_carre = x*x + y*y
```

```
26          if distance_au_carre > cote**2:
27              uneTortue.color('red')
28          else:
29              uneTortue.color('blue')
30              points_interieurs = points_interieurs + 1
31          uneTortue.dot()
32          j = j + 1
33      return (points_interieurs/nombre_de_points)*4
34
35  caroline = turtle.Turtle()
36  caroline.shape('turtle')
37  caroline.speed("fastest")
38
39  turtle.tracer(0,0) # Can be used to accelerate the drawing of complex graphics
40
41  axes(caroline, 200)
42  print("valeur approchee de pi :", monte_carlo(caroline, 200, 3500))
```

3.3.7 —

3.3.8 —

3.3.9 —

Chapitre 4

Codes et autres secrets

4.1 Les nombres premiers

4.1.1 (Le crible d'Ératosthène)
Dans le tableau ci-dessous, souligner 2, puis biffer tous les multiples de 2. Le premier nombre non biffé est 3. Il est premier, on le souligne et on biffe tous ses multiples. Etc. Arrivé à 10, tous les nombres non biffés sont premiers.
Reproduire cette méthode dans le tableau ci-dessous.

	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

4.1.2

a) Parmi les nombres 1999, 2999, 3999 et 4999, déterminer avec une calculatrice ceux qui sont premiers.

b) Un nombre de Fermat est un nombre qui peut s'écrire sous la forme $F_n = 2^{2^n} + 1$, avec n un entier naturel non nul.
Calculer F_2, F_3 et F_4, puis déterminer avec une calculatrice ceux qui sont premiers.

c) Écrire en python une fonction `isPrime(n)` qui détermine si le nombre entier n est premier.

d) Calculer F_5, est-il premier ? Sinon, donner un de ses diviseurs. Idem avec F_6.

e) Un nombre de Mersenne est un nombre qui peut s'écrire sous la forme $M_n = 2^n - 1$, avec n un entier naturel non nul. Déterminer un diviseur de M_{67}.

f) Écrire en python une fonction `listPrime(n)` qui retourne la liste des nombres premiers inférieurs ou égaux à n.

4.1.3 Démontrer que tout nombre naturel n, distinct de 0 et de 1, admet au moins un facteur premier.

4.1.4 Démontrer que l'ensemble des nombres premiers est infini.

4.1.5 Montrer qu'il existe une infinité de nombres premiers de la forme $p = 4n + 3$ avec $n \in \mathbb{N}^\star$.

4.1.6 Soit p, q deux nombres premiers distincts. Quels sont les diviseurs positifs de $p^2\,q$?

4.1.7 Tout entier n peut se décomposer en produit de facteurs premiers (sauf 0 et 1).

$$n = {p_1}^{\alpha_1} \cdot {p_2}^{\alpha_2} \cdot ... \cdot {p_r}^{\alpha_r}$$

avec p_j des nombres premiers distincts et α_j des entiers naturels non nuls.
Avec une calculatrice, décomposer en facteurs premiers les nombres suivants 100, 1024, 6000 et M_{10}.

4.1.8 Écrire une fonction `dec(N)` qui factorise un entier N.
Par exemple :

```
>>> dec(1000)
    [2, 2, 2, 5, 5, 5]
```

4.1.9 (La répartition des nombres premiers)
Soit n un entier naturel non nul.
Démontrer qu'il existe un intervalle de taille n sans aucun nombre premier.

4.1.10 On pose $\alpha = 101\,! + 1$. Montrer que les 100 nombres consécutifs

$$\alpha + 1, \alpha + 2, ..., \alpha + 100$$

ne sont pas premiers.

4.1.11 Trouver la plus petite série de 100 nombres consécutifs dont aucun n'est premier.

4.1.12 Si p est premier, avec $p > 5$, alors 24 divise $p^2 - 1$.

4.2 Divisibilité et congruences

Pour a et b dans $\mathbb{Z}$, on dit que a *divise* b s'il existe $z \in \mathbb{Z}$ tel que $a \cdot z = b$. Si a divise b, on le note $a \mid b$ et on dira de plus que a est un *diviseur* de b, ou que b est un *multiple* de a, ou encore que b est *divisible* par a. Si a ne divise pas b, on le note $a \nmid b$.

4.2.1 Soit a, b et c dans $\mathbb{Z}$. Montrer que

a) $a \mid a$, $1 \mid a$ et $a \mid 0$;

b) $0 \mid a$ si et seulement si $a = 0$;

c) $a \mid b$ si et seulement si $-a \mid b$ si et seulement si $a \mid -b$;

d) $a \mid b$ et $a \mid c$ implique $a \mid (b + c)$;

e) $a \mid b$ et $b \mid c$ implique $a \mid c$.

4.2.2
Soit $m, n, r \in \mathbb{Z}$. Montrer que

a) si $m \mid n$ et $m \mid r$, alors $m \mid (n - r)$;

b) si $m \mid n$ alors $m \mid r\,n$;

c) si $r \neq 0$, alors $m \mid n$ si et seulement si $r\,m \mid r\,n$.

4.2.3 Quels sont les diviseurs de 0 ?

4.2.4 Soit a et b dans $\mathbb{Z}$. Montrer que si $b \neq 0$ et que $a \mid b$, alors $1 \leq |a| \leq |b|$.

4.2.5 Soit a, b dans $\mathbb{Z}$. Montrer que $a \mid b$ et $b \mid a$ si et seulement si $a = \pm b$.

4.2.6 Trouver toutes les valeurs de $a \in \mathbb{Z}$, sachant que $a \mid 1$.

4.2.7 Soit a, b dans $\mathbb{Z}$, avec $b > 0$. Montrer qu'il existe q, r dans $\mathbb{Z}$, uniques, tels que

$$a = b\,q + r \quad \text{et} \quad 0 \leq r < b$$

4.2.8 Effectuer la division en colonnes, avec reste.

a) $13 \div 5$ c) $24 \div 13$ e) $100 \div 17$ g) $45 \div 8$

b) $25 \div 8$ d) $10 \div 3$ f) $26 \div 17$ h) $50 \div 10$

4.2.9 Quels sont tous les restes possibles après division entière par 3 ? Donner la liste de ces restes.

4.2.10 Quels sont tous les restes possibles après division entière par 26 ? Donner la liste de ces restes.

4.2.11 Quels sont les nombres dont le reste après division par 26 donne 17 ? Peut-on en donner la liste complète ?

4.2.12 On donne un nombre naturel a. Chercher son reste après division par n.

a) $a = 111$; $n = 2, 3, 4, \ldots, 12$;

b) $a = 123456789 \cdot 987654321$; $n = 2$, $n = 9$, $n = 11$;

c) $a = 22^{22}$, $n = 3$, $n = 9$, $n = 10$, $n = 11$;

d) $a = 1234^5$, $n = 9$, $n = 11$, $n = 99$;

e) $a = 2^{64} - 1$, $= 2$, $n = 3$, $n = 9$.

Soit x un nombre réel. On note $\lfloor x \rfloor$ la *partie entière* de x, qui désigne le plus grand entier m tel que $m \leq x$. De manière équivalente, $\lfloor x \rfloor$ est l'unique entier tel que $m \leq x < m + 1$. C'est également le seul nombre entier tel que $x = m + \varepsilon$ avec $0 \leq \varepsilon < 1$.

4.2.13 Donner la partie entière des nombre ci-dessous.

a) 1.5

b) -3.4

c) 0.001

d) $-\sqrt{2}$

e) $15/7$

f) π

g) $\sqrt{3}/2$

Soit a, b dans $\mathbb{Z}$ avec $b > 0$. Soit encore q et r les deux entiers uniques qui satisfont

$$a = b\,q + r \quad \text{et} \quad 0 \leq r < b$$

On définit

$$a \mod b = r$$

Autrement dit, $a \mod b$ désigne le reste de la division de a par b.

4.2.14 Soit a, b dans $\mathbb{Z}$ avec $b > 0$. Montrer que $b \mid a$ si et seulement si $a \mod b = 0$.

4.2.15 Soit a, b dans $\mathbb{Z}$ avec $b > 0$. Montrer que

$$a \mod b = a - b\,\lfloor a/b \rfloor$$

4.2.16 Donner une définition de $a \mod b$ pour a et b deux entiers quelconques.

4.2.17 Calculer, après avoir déterminé la meilleure méthode pour ce faire à l'aide de la calculette.

a) $53 \mod 12$

b) $-187 \mod 7$

c) $2333 \mod 428$

d) $-523 \mod 13$

e) $-100 \mod -15$

f) $1023 \mod -117$

g) $1341234123411237 \mod 2341234347$

Vérifier les résultats à l'aide de l'opérateur `%` de python.

4.2.18 Soit a, b dans $\mathbb{Z}$ avec $b < 0$. Montrer que

$$(a \mod b) \in]b, 0]$$

4.2.19 Soit n un entier positif. Soit encore a et b deux entiers quelconques.
On dit que a est *congru* à b *modulo* n si $n \mid (a - b)$, ce que nous écrivons $a \equiv b \pmod{n}$.
Montrer que cela revient à demander que $a = b + n \cdot y$ pour $y \in \mathbb{Z}$.

4.2.20 Soit n un entier positif. Soit encore a, b, c et d des entiers quelconques. Montrer que

a) $a \equiv a \pmod{n}$;
b) si $a \equiv b \pmod{n}$, alors $b \equiv a \pmod{n}$;
c) si $a \equiv b \pmod{n}$ et $b \equiv c \pmod{n}$, alors $a \equiv c \pmod{n}$;
d) tout nombre a est congru modulo n au reste de la division de a par n ;
e) on a $a \equiv b \pmod{n}$ si et seulement si a et b ont le même reste après division par n ;
f) si $a \equiv c \pmod{n}$ et si $b \equiv d \pmod{n}$, alors $a + b \equiv c + d \pmod{n}$ et $a \cdot b \equiv c \cdot d \pmod{n}$.

4.2.21 Soit n un entier positif et a et b deux entiers quelconques.

a) Démontrer que

$$(a + b) \mod n = ((a \mod n) + (b \mod n)) \mod n$$

b) Démontrer que

$$(a \cdot b) \mod n = ((a \mod n) \cdot (b \mod n)) \mod n$$

4.2.22 Trouver une dizaine de nombres entiers congrus à 22 modulo 7.

4.2.23 On calcule *modulo 6*, c'est à dire avec les entiers compris entre 0 et 5. L'ensemble de ces nombres peut être noté

$$\mathbb{Z}_5 = \{0, 1, 2, 3, 4, 5\}.$$

Cela revient à effectuer les opérations habituelles, en réduisant modulo 6 le résultat. En python, on peut utiliser l'opérateur `%`, comme sur les exemples ci-dessous :

```
>>> 3 + 4
>>> 7
>>> 7 % 6
>>> 1
```

Le calcul fait dans la console montre que

$$(3 + 4) \mod 6 = 1$$

Établir la table d'addition de $\mathbb{Z}_6$. Faire de même pour la table de multiplication.

4.2.24 On calcule *modulo 5*, c'est à dire avec les entiers compris entre 0 et 4. L'ensemble de ces nombres peut être noté

$$\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}.$$

Établir la table d'addition de $\mathbb{Z}_5$. Faire de même pour la table de multiplication.

4.2.25 On calcule *modulo 3*, c'est à dire avec les entiers compris entre 0 et 2. L'ensemble de ces nombres peut être noté

$$\mathbb{Z}_3 = \{0, 1, 2\}.$$

Établir la table de multiplication de $\mathbb{Z}_3$.

4.2.26 En observant les tables de multiplication et d'addition de $\mathbb{Z}_6$, on constate, entre autres que

a) $3 + 3 = 0$ modulo 6 ;

b) $2 \cdot 3 = 0$ modulo 6.

Que peut-on en déduire ?

4.2.27 On dit que l'on *additionne modulo 26* si l'on remplace le résultat de l'addition de deux nombres entiers par le reste de ce résultat après division par 26, si nécessaire. Donner le résultat modulo 26 des calculs ci-dessous.

a) $12 + 3$
b) $7 + 15$
c) $11 + 11$
d) $15 + 8$
e) $13 + 13$
f) $5 + 20$
g) $21 + 22$
h) $17 + 16$
i) $24 + 3$
j) $21 + 7$
k) $8 + 19$
l) $5 + 25$
m) $12 - 15$
n) $20 - 25$
o) $8 - 19$
p) $5 - 25$

4.2.28 Effectuer les calculs ci-dessous et réduire le résultat modulo 26.

a) $27 + 33$
b) $41 + 28$
c) $35 + 36$
d) $72 + 100$
e) $132 - 150$
f) $53 - 12$
g) $30 - 8$
h) $-5 - 12$

4.2.29 Pour chaque calcul donné à l'exercice 4.2.28, réduire les nombres modulo 26 avant d'effectuer l'opération. On veillera à réduire le résultat modulo 26 également, après avoir fait le calcul. Cela modifie-t-il le résultat final ?

4.2.30 Effectuer les calculs ci-dessous et réduire le résultat modulo 26.

a) $6 \cdot 3$
b) $5 \cdot 4$
c) $2 \cdot 12$
d) $8 \cdot 4$
e) $5 \cdot 6$
f) $10 \cdot 4$
g) $15 \cdot 3$
h) $5 \cdot 7$
i) $12 \cdot 3$
j) $7 \cdot 6$
k) $8 \cdot 9$
l) $20 \cdot 30$
m) $14 \cdot 5$
n) $25 \cdot 2$
o) $4 \cdot 11$
p) $6 \cdot 6$

4.2.31 Effectuer les calculs ci-dessous et réduire le résultat modulo 26.

a) $27 \cdot 33$

b) $41 \cdot 28$

c) $35 \cdot 36$

d) $72 \cdot 100$

e) $132 \cdot 150$

f) $53 \cdot 12$

g) $30 \cdot 8$

h) $(-5) \cdot 12$

4.2.32 Pour chaque calcul donné à l'exercice 4.2.31, réduire les nombres modulo 26 avant d'effectuer l'opération. On veillera à réduire le résultat modulo 26 également, après avoir fait le calcul. Cela modifie-t-il le résultat final ?

4.2.33 Donner tous les nombres qui sont dans l'ensemble $\mathbb{Z}_{26}$. Faire calculer la table de multiplication de $\mathbb{Z}_{26}$. Dans un tableur, mettre en évidence les résultats 0 et 1.

Écrire l'ensemble des inversibles de $\mathbb{Z}_{26}$, noté $\mathbb{Z}_{26}^*$.

4.2.34 Résoudre les trois équations modulaires suivantes.

a) $4\,x \equiv 2 \pmod 7$ b) $6\,x \equiv 9 \pmod{15}$ c) $4\,x \equiv 5 \pmod{14}$

4.2.35 Résoudre les quatre équations modulaires suivantes.

a) $5\,x + 6 \equiv 13 \pmod{26}$

b) $9\,x + 11 \equiv 3 \pmod{26}$

c) $16\,x + 18 \equiv 13 \pmod{26}$

d) $16\,x + 17 \equiv 13 \pmod{26}$

4.2.36

a) L'entier 6 a t-il un inverse modulo 77 ? Si oui, en déterminer un.

b) Résoudre $6\,x \equiv 5 \pmod{77}$

4.3 Le chiffre de César

4.3.1 On numérote les lettres majuscules de l'alphabet à partir de 0. Traduire le message

`ILYACEUXQUIONTUN`

à l'aide d'une liste de nombres de $\mathbb{Z}_{26}$.

4.3.2 On donne la liste de nombres appartenant à $\mathbb{Z}_{26}$

$$M = (4, 19, 2, 4, 20, 23, 16, 20, 8, 2, 17, 4, 20, 18, 4, 13, 19)$$

Trouver le message correspondant à M.

4.3.3 Chiffrer le message M de l'exercice 4.3.2 à l'aide du code de César. Donner le chiffre sous forme d'une liste de nombres.

4.3.4 Chiffrer le message

`ILYACEUXQUIONTUNPISTOLETCHARGE`

en utilisant le code de César.

4.3.5 Chiffrer, à l'aide du décalage de César, le message suivant :

`PARFOISMAVIEOUVRAITLESYEUXDANSLOBSCURITE`

en utilisant le code de César.

4.3.6 J'ai chiffré un message avec le code de César et le chiffre résultant est

`HWFHXATXLFUHXVHQW`

Quel était le message secret ?

4.3.7 Chiffrer le message donné ci-dessous à l'aide du décalage 7 :

$$M = (19, 14, 8, 19, 20)$$

Donner le résultat sous forme d'une liste de nombres.

Plutôt que dire que le décalage vaut 7, on dira que la *clef de chiffrement* vaut 7.

4.3.8 Trouver une technique pour retrouver le message M à partir du chiffre

$$C = (9, 24, 11, 1, 25, 11, 25)$$

sachant qu'il a été codé à l'aide d'un décalage de 7.

4.3.9 Combien y a-t-il de décalages de « type César » possibles ?

4.3.10 Donner une technique générale pour déchiffrer un message codé à l'aide d'un décalage de César quelconque. Si l'on note $n \in \mathbb{Z}_{26}$ le décalage utilisé, donner une formule qui donne le décalage permettant de déchiffrer le message ou *clef de déchiffrement*.

4.3.11 J'ai chiffré un message avec le décalage 13. Trouver la clef de déchiffrement.

4.3.12 Chiffrer le message m ci-dessous à l'aide du chiffrement de César et la clef 24.

```
JAMAISPERSONNENAVAITETEAUSSITROPFORT
```

Donner toutes les étapes du chiffrement :

a) écrire M, la liste des nombres qui correspondent aux lettres du message ;

b) chiffrer M, c'est à dire donner la liste C des nombres calculés à l'aide de la formule

$$w = (z + 24) \mod 26;$$

c) écrire c le chiffre formé de la suite des lettres qui correspondent aux nombres de C.

4.3.13 On donne ci-dessous le chiffre c obtenu à l'aide du chiffrement de César et la clef 4 :

```
PEXSTMWWMXYHIIXPEWIHYGXMZMXI
```

Donner toutes les étapes du déchiffrement :

a) écrire C, la liste des nombres qui correspondent aux lettres du message ;

b) déchiffrer C, c'est à dire donner la liste M des nombres calculés à l'aide de la formule

$$z = (w + (26 - 4)) \mod 26;$$

c) écrire m le message formé de la suite des lettres qui correspondent aux nombres de M.

4.3.14 À l'aide d'un ordinateur et en utilisant python, chiffrer les messages ci-dessous avec le décalage de César indiqué.

```
LAFOLLAANONIMACHERENDEANONIMIQUASIINVISIBILICOSICOSA 2
MAITRECORBEAUSURUNARBREPERCHETENAITENSONBECUNFROMAGE 15
ISURVIVEDANOTHERMEETINGTHATSHOUDHAVEBEENANEMAIL 7
IFYOUDONTKNOWTHATYOUDONTKNOWYOUTHINKYOUKNOW 5
QUOUSQUETANDEMABUTERECATILINAPATIENTIANOSTRA 14
UNCHAMPDEBLEPRENAITRACINESOUSLACOIFFEDEBECASSINE 2
VOMKLEINENMAULWURFDERWISSENWOLLTEWER 13
QUOILAMICECROCESTALAMODEPOURPRENDRESONCHAPEAU 8
TOBEORNOTTOBETHATISTHEQUESTIONWHETHERTISNOBLERINTHEMIND 3
ONVAALLERALHOTELSANSPAYERONVASINSCRIREALAFACSANSYALLER 21
```

4.3.15 À l'aide d'un ordinateur et en utilisant python, déchiffrer les messages ci-dessous avec le décalage de César indiqué.

```
SGOZXKIUXHKGAYAXATGXHXKVKXINKZKTGOZK  chiffré avec le décalage  6
VKPYVVKKXYXSWKMROBOXNOKXYXSWSAEKCSSX  chiffré avec le décalage  10
QNGWCLWVBSVWEBPIBGWCLWVBSVWEGWCBPQVS  chiffré avec le décalage  8
BLNKOBOXWTGHMAXKFXXMBGZMATMLAHNWATOX  chiffré avec le décalage  19
YCWQTIUQKMKZWKMABITIUWLMXWCZXZMVLZMA  chiffré avec le décalage  8
JNHNLJNXMTGWXFTUNMXKXVTMBEBGTITMBXGM  chiffré avec le décalage  19
SNADNQMNSSNADSGZSHRSGDPTDRSHNMVGDSGD  chiffré avec le décalage  25
ATINGSVJKHRKVXKTGOZXGIOTKYUAYRGIUOLL  chiffré avec le décalage  6
DCKPPAATGPAWDITAHPCHEPNTGDCKPHXCHRGX  chiffré avec le décalage  15
NGECDWAFWFESMDOMJXVWJOAKKWFOGDDLWOWJ  chiffré avec le décalage  18
```

4.3.16 On a pu intercepter le chiffre ci-dessous :

```
ARZTLVZCCZTVSIZEUVSILPVIV
```

On sait que l'expéditeur a chiffré le message clair à l'aide du code de César, mais on ignore la clef. Comment faire pour trouver le message clair ? Cela revient à *casser le code de César*.

4.3.17 On a pu intercepter le chiffre ci-dessous :

```
SNBDRBUJVJDEJRBNQNAKNKAJENBPNWBKAJENBPNWB
```

Trouver le texte clair.

4.3.18 En utilisant le fichier liste_mots_francais.txt, écrire un programme qui casse le chiffre de césar automatiquement pour un message d'origine écrit en français. Le programme retourne le message clair et la clef de chiffrement.

4.3.19 Le `ROT13` (de l'anglais rotate by 13 places) est un cas particulier du chiffre de César. Comme son nom l'indique, il s'agit de décaler chaque lettre du texte à chiffrer de 13 places.
Quelle est sa particularité ?

4.4 Le chiffrement affine

4.4.1 Établir la table de la fonction $f(z) = (2 \cdot z + 3) \mod 5$. Peut-on inverser cette fonction ? Justifier.

4.4.2 Établir la table de la fonction $f(z) = (2 \cdot z + 3) \mod 4$. Peut-on inverser cette fonction ? Justifier.

4.4.3 Soit z un nombre de $\mathbb{Z}_{26}$. On a vu que z peut représenter une lettre de l'alphabet majuscule. On sait que pour le chiffrement de type César on choisit une clef n et que le chiffrement de z est calculé à l'aide de la formule

$$w = (z + n) \mod 26$$

On peut définir une autre méthode de chiffrement, nommée *chiffrement affine* en suivant la procédure ci-dessous :

a) On choisit a et b deux nombres de Z_{26}, avec a inversible modulo 26.

b) On chiffre une lettre majuscule de l'alphabet, représentée par un nombre entier modulo 26, de la façon suivante :

$$w = (a \cdot z + b) \mod 26$$

c) Trouver une formule permettant de retrouver z à partir de w, a, et b.

Chiffrer le message ci-dessous à l'aide du chiffrement affine avec $a = 5$ et $b = 3$.

`JAMAISPERSONNENAVAITETEAUSSITROPFORT`

Quelle est la fonction de déchiffrement ? Retrouver le message à partir du chiffre.

4.4.4 Chiffrer, à l'aide du décalage affine dont la clef est $(3, 13)$, le message suivant :

`PARFOISMAVIEOUVRAITLESYEUXDANSLOBSCURITE`

Donner la fonction de déchiffrement et retrouver le message à partir du chiffre.

4.4.5 On a chiffré un message à l'aide du chiffrement affine avec $a = 15$ et $b = 7$. Le chiffre est donné ci-dessous :

`QHGJYXRRXGVAPPGQHRPAVLGXKXGP`

Déchiffrer ce message.

4.4.6 À l'aide d'un ordinateur et en utilisant python, chiffrer les messages ci-dessous avec le chiffrement affine en utilisant la clef indiquée.

```
LAFOLLAANONIMACHERENDEANONIMIQUASIINVISIBILICOSICOSA (17, 24)
MAITRECORBEAUSURUNARBREPERCHETENAITENSONBECUNFROMAGE (15, 15)
ISURVIVEDANOTHERMEETINGTHATSHOUDHAVEBEENANEMAIL (3, 7)
IFYOUDONTKNOWTHATYOUDONTKNOWYOUTHINKYOUKNOW (5, 12)
QUOUSQUETANDEMABUTERECATILINAPATIENTIANOSTRA (11, 2)
UNCHAMPDEBLEPRENAITRACINESOUSLACOIFFEDEBECASSINE (7, 3)
VOMKLEINENMAULWURFDERWISSENWOLLTEWER (1, 1)
QUOILAMICECROCESTALAMODEPOURPRENDRESONCHAPEAU (19, 9)
TOBEORNOTTOBETHATISTHEQUESTIONWHETHERTISNOBLERINTHEMIND (21, 22)
ONVAALLERALHOTELSANSPAYERONVASINSCRIREALAFACSANSYALLER (23, 2)
```

4.4.7 À l'aide d'un ordinateur et en utilisant python, déchiffrer les messages ci-dessous avec la clef de chiffrement affine indiquée.

```
SIKMLGCMOWOJKOWARGLGCKFWBKIJBJWTFJWA  avec la clef (17, 6)
WBEMCNMFVKFMQVPSVEMCNMFVKFMQEMCVPWFK  avec la clef (7, 18)
CTQPGODVALJADNATGUXNGQUTASYCSJGQYUFF  avec la clef (5, 6)
QTFRYBDMBMFZJYXJOIUBOXDVVBMXTYYCBXBO  avec la clef (7, 25)
OKENBOBADMZWHRANCAAHOZUHRMHKRWEDRMBA  avec la clef (23, 12)
VNBEUHXFUSHNJZJUJANUSUHKHUXWHEHANBEH  avec la clef (5, 13)
RKXWKJTKRRKXWRVGRMARVWSIWARMKTQVWRVW  avec la clef (17, 6)
IOSOYIOQTKXVQCKFOTQDQAKTWHWXKNKTWQXT  avec la clef (21, 10)
NGVWNNGGTWTEQGMBSFSTPSGTWTEQECOGIEET  avec la clef (3, 6)
ARLEEZZOBEZPATOZKERKJEMOBARLEKYRKWBY  avec la clef (9, 4)
```

4.4.8 En utilisant le fichier liste_mots_francais.txt, écrire un programme qui casse le chiffre affine automatiquement pour un message d'origine écrit en français. Le programme retourne le message clair et la clef de chiffrement.

4.5 Solutions des exercices

Les nombres premiers

4.1.1

Les nombres premiers inférieurs à 100 : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.

4.1.2

a) 1999, 2999 et 4999 sont premiers.

b) $F_2 = 17$, $F_3 = 257$ et $F_4 = 65537$. Ils sont premiers.

c)

```
def isPrime(n):
    for k in range(2,int(n**0.5)+1):
        if n%k == 0:
            return False
    return True
```

d) $F_5 = 4294967297$, il n'est pas premier $F_5 = 641 \cdot 6700417$.

e) $M_{67} = 2^{67} - 1 = 147573952589676412927 = 193707721 \cdot 761838257287$

f)

```
def listPrime(n):
    l = []
    for k in range(2, n+1):
        if isPrime(k):
            l.append(k)
    return l
```

4.1.3

Faisons un raisonnement par l'absurde.

Si n n'est pas premier, alors n admet au moins un diviseur inférieur à n et strictement plus grand que 1.

Soit p le plus petit de ces diviseurs. On a $1 < p < n$.

Supposons que p ne soit pas un nombre premier. Alors il existe $d \in \mathbb{N}$ tel que $1 < d < p$ et d divise p. Alors d divise n, ce qui est impossible.

p est donc un nombre premier.

4.1.4 –

4.1.5

L'ensemble des nombres premiers est infini. Soit $p > 2$ premier, alors p est impair. Donc il s'écrit $p = 2\,n + 1$, avec $n \in \mathbb{N}^\star$.

Si n est impair, il s'écrit $n = 2\,k + 1$, avec $k \in \mathbb{N}^\star$. Donc $p = 2\,(2\,k + 1) + 1 = 4\,k + 3$.

Il existe donc une infinité de nombres premiers de la forme $4\,k + 3$.

4.1.6

$p, q, p^2, p\,q, p^2\,q$

4.1.7 –

4.1.8

```
def dec(N):
    Resultat = []
    d = 2
    while N%d == 0:
        Resultat.append(d)
        N = N//d
    d = 3
    while d <= N:
        while N%d == 0:
            Resultat.append(d)
            N = N//d
        d = d + 2
    return Resultat
```

4.1.9 –

4.1.10 –

4.1.11

```
def isPrime(n):
    for k in range(2,int(n**0.5)+1):
        if n%k == 0:
            return False
    return True

p = 3
q = 5

while q - p < 100:
    p, q = q, q + 2
    while isPrime(q) == False:
        q = q + 2

print(p, q)
```

370262 à 370362

4.1.12

On a $p^2 - 1 = (p-1)(p+1)$.

Comme $p > 5$ et est impair, on a que $p-1$ et $p+1$ sont pairs.

$p-1$, p et $p+1$ sont consécutifs. Donc l'un des 3 est un multiple de 3, et ce n'est pas p

De même, l'un des 3 est un multiple de 4, sinon $p-2$ est un multiple de 4 et donc p serait pair.

Divisibilité et congruences

4.2.1 —

4.2.2 —

4.2.3 Tout $z \in \mathbb{Z}$ est un diviseur de 0.

4.2.4 —

4.2.5 —

4.2.6 $a = \pm 1$

4.2.7 —

4.2.8

a) 2 reste 3

b) 3 reste 1

c) 1 reste 11

d) 3 reste 1

e) 5 reste 15

f) 1 reste 9

g) 5 reste 5

h) 5 reste 0

4.2.9 $(0, 1, 2)$

4.2.10 $(0, 1, 2, 3, 4, 5, \ldots, 24, 25)$

4.2.11 Ce sont les nombres de la forme $17 + k \cdot 26$. On ne peut pas donner explicitement la liste de tous ces nombres, vu qu'il y en a une infinité.

4.2.12 Le code ci-dessous fournit les solutions :

```
liste_a = [111,
           123456789 * 987654321,
           22**22,
           1234**5,
           2**64-1]
listes_n = [[2,3,4,12],
            [2,9,11],
            [3,9,10,11],
            [9,11,99],
            [2,3,9]]
k = len(liste_a)
for i in range(k):
    a = liste_a[i]
    nombres = listes_n[i]
    for n in nombres:
        print(f"{a} mod {n} = {a%n}")
    print()
```

Après exécution, on obtient l'affichage suivant à la console :

```
111 mod 2 = 1
111 mod 3 = 0
111 mod 4 = 3
111 mod 12 = 3

121932631112635269 mod 2 = 1
121932631112635269 mod 9 = 0
121932631112635269 mod 11 = 3

341427877364219557396646723584 mod 3 = 1
341427877364219557396646723584 mod 9 = 4
```

```
341427877364219557396646723584 mod 10 = 4
341427877364219557396646723584 mod 11 = 0


2861381721051424 mod 9 = 1
2861381721051424 mod 11 = 10
2861381721051424 mod 99 = 10


18446744073709551615 mod 2 = 1
18446744073709551615 mod 3 = 0
18446744073709551615 mod 9 = 6
```

4.2.13

a) 1

b) -4

c) 0

d) -2

e) 2

f) 3

g) 0

4.2.14 —

4.2.15 —

4.2.16 On pose $a \mod b = a - b\lfloor a/b \rfloor$, qui est défini pour tout $a, b \in \mathbb{Z}$.

4.2.17 Pour calculer $1234 \mod 123$, par exemple, on effectue la division avec la calculette, qui donne

$$1234 \div 123 = 10.03252033$$

On soustrait du résultat sa partie entière, ce qui revient à faire le calcul

$$10.03252033 - 10 = 0.03252033$$

Il suffit alors de multiplier le dernier résultat par le diviseur pour trouver le reste :

$$0.03252033 \cdot 123 = 4$$

En règle générale, l'algorithme de multiplication de la calculette donne la valeur exacte. On doit bien entendu adapter cette technique aux cas dans lesquels les nombres ne sont pas tous deux positifs.

a) $53 \div 12 \simeq 4.416666667$, $4.416666667 - 4 = 0.416666667$, $12 \cdot 0.416666667 = 5$

b) $-187 \div 7 \simeq -26.71428571$, $-26.71428571 + 27 = 0.285714286$, $7 \cdot 0.285714286 = 2$

c) $2333 \div 428 \simeq 5.450934579$, $428 \cdot 0.450934579 = 193$

d) $-523 \div 13 \simeq -40.23076923$, $-40.23076923 + 41 = 0.769230769$, $13 \cdot 0.769230769 = 10$

e) $-100 \mod -15 = -100 - (-15) \cdot \lfloor -100/(-15) \rfloor = -100 + 15 \cdot \lfloor 6.6666\ldots \rfloor$, ce qui donne : $-100 + 90 = -10$

f) $1023 \mod -117 = 1023 - (-117) \cdot \lfloor 1023/(-117) \rfloor = 1023 + 117 \cdot (-9) = -30$

g) La calculette ordinaire ne suffit plus. le résultat est 1838107959.

4.2.18 —

4.2.19 —

4.2.20 —

4.2.21 —

4.2.22 Il suffit de choisir parmi les nombres de la famille définie par $22 + k \cdot 7$ pour $k \in \mathbb{Z}$.

4.2.23 Dans le tableau t ci-dessous, la case t_{nm} est définie par

$$t_{nm} = (n + m) \mod 6$$

```
0 1 2 3 4 5
1 2 3 4 5 0
2 3 4 5 0 1
3 4 5 0 1 2
4 5 0 1 2 3
5 0 1 2 3 4
```

Dans le tableau t ci-dessous, la case t_{nm} est définie par

$$t_{nm} = (n \cdot m) \mod 6$$

```
0 0 0 0 0 0
0 1 2 3 4 5
0 2 4 0 2 4
0 3 0 3 0 3
0 4 2 0 4 2
0 5 4 3 2 1
```

4.2.24 Dans le tableau t ci-dessous, la case t_{nm} est définie par

$$t_{nm} = (n + m) \mod 5$$

```
0 1 2 3 4
1 2 3 4 0
2 3 4 0 1
3 4 0 1 2
4 0 1 2 3
```

Dans le tableau t ci-dessous, la case t_{nm} est définie par

$$t_{nm} = (n \cdot m) \mod 5$$

```
0 0 0 0 0
0 1 2 3 4
0 2 4 1 3
0 3 1 4 2
0 4 3 2 1
```

4.2.25 Dans le tableau t ci-dessous, la case t_{nm} est définie par

$$t_{nm} = n \cdot m \mod 3$$

```
0 0 0
0 1 2
0 2 1
```

4.2.26 On peut dire que 3 est l'opposé de 3 modulo 6.

On peut en déduire que 2 et 3 n'ont pas d'inverse modulo 6.

4.2.27

a) 15 e) 0 i) 27 m) 23

b) 22 f) 25 j) 2 n) 21

c) 22 g) 17 k) 1 o) 15

d) 23 h) 7 l) 4 p) 6

4.2.28

a) 8 c) 19 e) 8 g) 22

b) 17 d) 16 f) 15 h) 9

4.2.29

a) $1 + 7 = 8$

b) $15 + 2 = 17$

c) $9 + 10 = 19$

d) $20 + 22 = 42 \equiv 16 \mod 26$

e) $2 - 20 = -18 \equiv 8 \mod 26$

f) $1 - 12 = -11 \equiv 15 \mod 26$

g) $4 + 18 = 22$

h) $21 + 14 = 35 \equiv 9 \mod 26$

Le résultat final n'est pas modifié.

4.2.30

a) 18 e) 4 i) 10 m) 18

b) 20 f) 24 j) 16 n) 24

c) 24 g) 19 k) 20 o) 18

d) 6 h) 9 l) 2 p) 10

4.2.31

a) $27 \cdot 33 = 891 \equiv 7 \mod 26$, vu que $891 = 34 \cdot 26 + 7$

b) $41 \cdot 28 = 1148 \equiv 4 \mod 26$, vu que $1148 = 44 \cdot 26 + 4$

c) $72 \cdot 100 = 7200 \equiv 24 \mod 26$, vu que $7200 = 276 \cdot 26 + 24$

d) $132 \cdot 150 = 19800 \equiv 14 \mod 26$, vu que $19800 = 761 \cdot 26 + 14$

e) $53 \cdot 12 = 636 \equiv 12 \mod 26$, vu que $636 = 24 \cdot 26 + 12$

f) $30 \cdot 8 = 240 \equiv 6 \mod 26$, vu que $240 = 10 \cdot 26 - 20$

g) $(-5) \cdot 12 = -60 \equiv 18 \mod 26$, vu que $-60 = (-2) \cdot 26 - 8$

4.2.32

a) $1 \cdot 7 = 7$

b) $15 \cdot 20 = 30 \equiv 4 \mod 26$

c) $9 \cdot 10 = 90 \equiv 12 \mod 26$

d) $20 \cdot 22 = 440 = 16 \cdot 26 + 24 \equiv 24 \mod 26$

e) $2 \cdot 20 = 40 \equiv 14 \mod 26$

f) $1 \cdot 12 = 12$

g) $4 \cdot 8 = 32 \equiv 6 \mod 26$

h) $21 \cdot 12 = 252 = 9 \cdot 26 + 18 \equiv 18 \mod 26$

Le résultat final n'est pas modifié.

4.2.33 L'ensemble cherché est

$$\mathbb{Z}_{26} = \{0, 1, 2, 3, 4, 5, \ldots, 20, 21, 22, 23, 24, 25\}$$

Le code ci-dessous fait afficher la table de multiplication de $\mathbb{Z}_{26}$.

```
k = 26
table = ""
```

```
for n in range(k):
    ligne = ""
    for m in range(k):
        ligne += str((m * n) % k).rjust(3)
    table += ligne
    table += "\n"
table = table[:-1]
print(table)
```

En contemplant le tableau affiché par le programme précédent ou son équivalent dans un tableur, on peut conclure que

$$\mathbb{Z}_{26}^* = \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$$

4.2.34

a) $x \equiv 4 \pmod 7$ b) $x \equiv 4 \pmod 5$ c) pas de solution

4.2.35

a) $x \equiv 17 \pmod{26}$

b) $x \equiv 2 \pmod{26}$

c) pas de solution

d) $x \equiv 3 \pmod{26}$ et $x \equiv 16 \pmod{26}$

4.2.36

a) 13 est l'inverse de 6 modulo 77.

b) $x = 65$

4.3.1 $(8, 11, 24, 0, 2, 4, 20, 23, 16, 20, 8, 14, 13, 19, 20, 13])$

4.3.2 `ETCEUXQUICREUSENT`

4.3.3 `[7, 22, 5, 7, 23, 0, 19, 23, 11, 5, 20, 7, 23, 21, 7, 16, 22]`

4.3.4 `LOBDFHXATXLRQWXQSLVWROHWFKDUJH`

4.3.5 `SDUIRLVPDYLHRXYUDLWOHVBHXAGDQVOREVFXULWH`

4.3.6 ETCEUXQUICREUSENT

4.3.7 $C = (0, 21, 15, 0, 1)$

4.3.8 On peut ajouter 19 et réduire modulo 7. Vu que $7 + 19 \equiv 0 \mod 26$, cela permet de retrouver la liste d'origine : $M = (2, 17, 4, 20, 18, 4, 18)$

4.3.9 Il y a 25 décalages si l'on excepte le « décalage 0 ».

4.3.10 $26 - n$

4.3.11 $26 - 13 = 13$

4.3.12 HYKYGQNCPQMLLCLYTYGRCRCYSQQGRPMNDMPR

4.3.13 LATOPISSITUDEETLASEDUCTIVITE

4.3.14

```
def cesar(texte, d):
    chiffre = ""
    for c in texte:
        n = ord(c)
        n = n - 65
        n = (n + d) % 26
        n = n + 65
        c_ = chr(n)
        chiffre = chiffre + c_
    return chiffre

messages = [("LAFOLLAANONIMACHERENDEANONIMIQUASIINVISIBILICOSICOSA", 2),
            ("MAITRECORBEAUSURUNARBREPERCHETENAITENSONBECUNFROMAGE", 15),
            ("ISURVIVEDANOTHERMEETINGTHATSHOUDHAVEBEENANEMAIL", 7),
            ("IFYOUDONTKNOWTHATYOUDONTKNOWYOUTHINKYOUKNOW", 5),
            ("QUOUSQUETANDEMABUTERECATILINAPATIENTIANOSTRA", 14),
```

```
            ("UNCHAMPDEBLEPRENAITRACINESOUSLACOIFFEDEBECASSINE", 2),
            ("VOMKLEINENMAULWURFDERWISSENWOLLTEWER", 13),
            ("QUOILAMICECROCESTALAMODEPOURPRENDRESONCHAPEAU", 8),
            ("TOBEORNOTTOBETHATISTHEQUESTIONWHETHERTISNOBLERINTHEMIND", 3),
            ("ONVAALLERALHOTELSANSPAYERONVASINSCRIREALAFACSANSYALLER", 21)]

for m in messages:
    print(cesar(m[0], m[1]))
```

```
NCHQNNCCPQPKOCEJGTGPFGCPQPKOKSWCUKKPXKUKDKNKEQUKEQUC
BPXIGTRDGQTPJHJGJCPGQGTETGRWTITCPXITCHDCQTRJCUGDBPVT
PZBYCPCLKHUVAOLYTLLAPUNAOHAZOVBKOHCLILLUHULTHPS
NKDTZITSYPSTBYMFYDTZITSYPSTBDTZYMNSPDTZPSTB
EICIGEISHOBRSAOPIHSFSQOHWZWBODOHWSBHWOBCGHFO
WPEJCORFGDNGRTGPCKVTCEKPGUQWUNCEQKHHGFGDGECUUKPG
IBZXYRVARAZNHYJHESQREJVFFRAJBYYGRJRE
YCWQTIUQKMKZWKMABITIUWLMXWCZXZMVLZMAWVKPIXMIC
WREHRUQRWWREHWKDWLVWKHTXHVWLRQZKHWKHUWLVQREOHULQWKHPLQG
JIQVVGGZMVGCJOZGNVINKVTZMJIQVNDINXMDMZVGVAVXNVINTVGGZM
```

4.3.15

```
MAITRECORBEAUSURUNARBREPERCHETENAITE
LAFOLLAANONIMACHERENDEANONIMIQUASIIN
IFYOUDONTKNOWTHATYOUDONTKNOWYOUTHINK
ISURVIVEDANOTHERMEETINGTHATSHOUDHAVE
QUOILAMICECROCESTALAMODEPOURPRENDRES
QUOUSQUETANDEMABUTERECATILINAPATIENT
TOBEORNOTTOBETHATISTHEQUESTIONWHETHE
UNCHAMPDEBLEPRENAITRACINESOUSLACOIFF
ONVAALLERALHOTELSANSPAYERONVASINSCRI
VOMKLEINENMAULWURFDERWISSENWOLLTEWER
```

4.3.16 On écrit une boucle qui teste toutes les clefs possibles et on sélectionne le message intelligible.

```
chiffre = "ARZTLVZCCZTVSIZEUVSILPVIV"


for i in range(1, 26):
    print(cesar(chiffre, i), 26 - i)
    print(50 * "*")
```

```
BSAUMWADDAUWTJAFVWTJMQWJW 25
**************************
CTBVNXBEEBVXUKBGWXUKNRXKX 24
**************************
DUCWOYCFFCWYVLCHXYVLOSYLY 23
**************************
EVDXPZDGGDXZWMDIYZWMPTZMZ 22
**************************
FWEYQAEHHEYAXNEJZAXNQUANA 21
**************************
GXFZRBFIIFZBYOFKABYORVBOB 20
**************************
HYGASCGJJGACZPGLBCZPSWCPC 19
**************************
IZHBTDHKKHBDAQHMCDAQTXDQD 18
**************************
JAICUEILLICEBRINDEBRUYERE 17
**************************
KBJDVFJMMJDFCSJOEFCSVZFSF 16
**************************
LCKEWGKNNKEGDTKPFGDTWAGTG 15
**************************
MDLFXHLOOLFHEULQGHEUXBHUH 14
```

```
************************
NEMGYIMPPMGIFVMRHIFVYCIVI 13
************************
OFNHZJNQQNHJGWNSIJGWZDJWJ 12
************************
PGOIAKORROIKHXOTJKHXAEKXK 11
************************
QHPJBLPSSPJLIYPUKLIYBFLYL 10
************************
RIQKCMQTTQKMJZQVLMJZCGMZM 9
************************
SJRLDNRUURLNKARWMNKADHNAN 8
************************
TKSMEOSVVSMOLBSXNOLBEIOBO 7
************************
ULTNFPTWWTNPMCTYOPMCFJPCP 6
************************
VMUOGQUXXUOQNDUZPQNDGKQDQ 5
************************
WNVPHRVYYVPROEVAQROEHLRER 4
************************
XOWQISWZZWQSPFWBRSPFIMSFS 3
************************
YPXRJTXAAXRTQGXCSTQGJNTGT 2
************************
ZQYSKUYBBYSURHYDTURHKOUHU 1
************************
```

Le chiffre avait été codé avec le décalage 17.

4.3.17 JESUISLAMAUVAISEHERBEBRAVESGENSBRAVESGENS

La clef de chiffrement est 9, celle de déchiffrement 17.

4.3.18 —

4.3.19 Son principal aspect pratique est que le codage et le décodage se font exactement de la même manière. En effet, puisque l'alphabet comporte 26 lettres, il suffit de décaler deux fois de 13 caractères pour revenir à la lettre de départ.

4.4.1

0	3
1	0
2	2
3	4
4	1

On voit qu'il s'agit d'une permutation des éléments de $\mathbb{Z}_5$. On peut donc inverser cette fonction et son inverse est la fonction $g(w) = (3 \cdot w - 4) \mod 5$.

4.4.2

0	3
1	1
2	3
3	1

On voit qu'il ne s'agit pas d'une permutation des éléments de $\mathbb{Z}_4$, vu que $f(0) = f(3) = 3$ et que $f(1) = f(3) = 1$. La fonction f n'admet pas de réciproque.

4.4.3 Il faut que a soit inversible modulo 26. Notons a^{-1} l'inverse de a modulo 26. On a

$$z = (w - b) \cdot a^{-1} \mod 26 \quad \Leftrightarrow \quad z = (a^{-1} \cdot w - a^{-1} \cdot b) \mod 26$$

```
def chiffrer_affine(message, a, b):
    chiffre = ""
    for lettre in message:
        z = ord(lettre) - 65
        z = (a*z + b) % 26
```

```
        c = chr(z + 65)
        chiffre = chiffre + c
    return chiffre

def dechiffrer_affine(chiffre, a, b):
    message = ""
    inverses = [1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25]
    a_ = 0
    for w in inverses:
        if (a*w)%26 == 1:
            a_ = w
    for lettre in chiffre:
        z = ord(lettre) - 65
        z = (a_*z - a_ * b) % 26
        c = chr(z + 65)
        message = message + c
    return message

chiffrer_affine("JAMAISPERSONNENAVAITETEAUSSITROPFORT", 5, 3)
'WDLDRPAXKPVQQXQDEDRUXUXDZPPRUKVACVKU'
dechiffrer_affine("WDLDRPAXKPVQQXQDEDRUXUXDZPPRUKVACVKU", 5, 3)
'JAMAISPERSONNENAVAITETEAUSSITROPFORT'
```

4.4.4

```
chiffrer_affine("PARFOISMAVIEOUVRAITLESYEUXDANSLOBSCURITE", 3, 13)
'GNMCDLPXNYLZDVYMNLSUZPHZVEWNAPUDQPTVMLSZ'
dechiffrer_affine("GNMCDLPXNYLZDVYMNLSUZPHZVEWNAPUDQPTVMLSZ", 3, 13)
'PARFOISMAVIEOUVRAITLESYEUXDANSLOBSCURITE'
```

4.4.5

```
dechiffrer_affine("QHGJYXRRXGVAPPGQHRPAVLGXKXGP", 15, 7)
'LATOPISSITUDEETLASEDUCTIVITE'
```

4.4.6

DYFCDDYYLCLEUYGNOBOLXOYLCLEUEKAYSEELRESEPEDEGCSEGCSY

NPFOKXTRKEXPDZDKDCPKEKXGXKTQXOXCPFOXCZRCEXTDCMKRNPBX

FJPGSFSTQHUXMCTGRTTMFUZMCHMJCXPQCHSTKTTUHUTRHFO

ALCEIBEZDKZESDVMDCEIBEZDKZESCEIDVAZKCEIKZES

WOAOSWOUDCPJUECNODUHUYCDMTMPCLCDMUPDMCPASDHC

NQRADJEYFKCFESFQDHGSDRHQFZXNZCDRXHMMFYFKFRDZZHQF

WPNLMFJOFONBVMXVSGEFSXJTTFOXPMMUFXFS

BZPFKJDFVHVUPVHNGJKJDPOHIPZUIUHWOUHNPWVMJIHJZ

FERCEPJEFFERCFNWFIKFNCUACKFIEJQNCFNCPFIKJERTCPIJFNCOIJH

MPRCCVVQDCVHMXQVACPAJCIQDMPRCAEPAWDEDQCVCNCWACPAICVVQD

4.4.7

QUOILAMICECROCESTALAMODEPOURPRENDRES

IFYOUDONTKNOWTHATYOUDONTKNOWYOUTHINK

UNCHAMPDEBLEPRENAITRACINESOUSLACOIFF

VOMKLEINENMAULWURFDERWISSENWOLLTEWER

ISURVIVEDANOTHERMEETINGTHATSHOUDHAVE

MAITRECORBEAUSURUNARBREPERCHETENAITE

TOBEORNOTTOBETHATISTHEQUESTIONWHETHE

QUOUSQUETANDEMABUTERECATILINAPATIENT

LAFOLLAANONIMACHERENDEANONIMIQUASIIN

ONVAALLERALHOTELSANSPAYERONVASINSCRI

4.4.8 —