Université de Nantes

BUT Info 1

Mathématiques discrètes

2021-2022

Travaux Dirigés

1 Logique

Exercice 1 Langage formel / langage courant

Soit p la proposition énoncée par Mathieu quant à son dernier déjeuner « J'ai mangé une boîte de pâtes (tu sais de celles que tu mets au micro-ondes et hop c'est prêt ?) » et q la proposition « J'ai mangé une salade de quinoa ». Traduire par une phrase en français :

1. $p \wedge q$
2. $p \vee q$
3. $\neg(\neg p \vee q)$

Exercice 2

On considère les propositions atomiques :
- P : « Ils ont bu du Perrier »,
- S : « Ils ont dansé la salsa »,
- C : « Ils ont écouté l'intégrale de Céline Dion »,
- H : « Ils étaient tous de bonne humeur »,
- I : « Leur soirée a été inoubliable »,
- T : « Leur soirée a été torride ».

2.1. Énoncez des phrases simples traduisant les propositions suivantes :

a. $(S \wedge T) \Rightarrow P$ **b.** $H \Rightarrow (I \Leftrightarrow C)$

2.2. Traduisez par une proposition simple les phrases :

a. « Ils n'étaient pas tous de bonne humeur s'ils n'ont pas dansé la salsa ».

b. « Ils ont pu être tous de bonne humeur seulement s'ils ont bu du Perrier ».

c. « Ils étaient tous de bonne humeur si et seulement si ils ont bu du Perrier et ont dansé la Salsa ».

d. « Il était suffisant d'écouter l'intégrale de Céline Dion pour que la soirée soit inoubliable ».

e. « Il était nécessaire d'écouter l'intégrale de Céline Dion pour que la soirée soit inoubliable ».

Exercice 3 Illustration de l'implication

3.1. On considère la proposition P : « Si une chose est un vélo, alors cette chose est bleue ».

a. Vous possédez un vélo bleu, est-ce que cela contredit P ?

b. Vous possédez une voiture rouge, est-ce que cela contredit P ?

c. Vous possédez une voiture bleue, est-ce que cela contredit P ?

d. Vous possédez un vélo vert, est-ce que cela contredit P ?

3.2. Pour chacune des situations précédentes, exprimez les valeurs de vérité de « cette chose est un vélo » et « cette chose est bleue ».
Présentez-le sous forme de tableau, avec la valeur de vérité de P associée.
Que reconnaissez-vous ?

Exercice 4

On considère la formule : $F_1 : (p \Rightarrow q) \wedge \neg r$.

4.1. Établir la table de vérité de F_1.

4.2. Est-ce que F_1 est une tautologie ?

4.3. Donner une valuation de p, q, r satisfaisant F_1.

4.4. Donner une valuation de p, q, r ne satisfaisant pas F_1.

4.5. En utilisant les règles de manipulation, exprimer F_1 sous forme normale disjonctive à l'aide des seuls connecteurs $\neg$, $\wedge$ et $\vee$.

Exercice 5

5.1. Comparer les deux formules suivantes :

- $(A \wedge B) \Rightarrow C$
- $A \Rightarrow (B \Rightarrow C)$

Exercice 6

6.1. Montrer les équivalences suivantes de deux manières (tables de vérité et manipulation de formules) :

a. $p \vee (p \wedge q) \equiv p$

b. $p \wedge (p \vee q) \equiv p$

c. $p \vee (\neg p \wedge q) \equiv p \vee q$

d. $(p \vee q) \wedge (p \vee \neg q) \equiv p$

e. $p \vee q \vee (\neg p \wedge \neg q) \equiv 1$

f. $(p \vee q) \wedge \neg p \wedge \neg q \equiv 0$

Exercice 7

7.1. Montrer que $\wedge$, $\vee$ et $\Leftrightarrow$ sont associatifs et commutatifs.

7.2. Montrer que $\wedge$ se distribue sur $\vee$ et inversement.

i.e. montrer que $a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$ *et* $a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$

7.3. Est-ce que $\Rightarrow$ est associative ? commutative ?

7.4. Est-ce que $\Rightarrow$ est transitive ? *i.e.* est-ce que si $p \Rightarrow q$ et $q \Rightarrow r$, alors $p \Rightarrow r$?

Exercice 8 Contraposée, réciproque et négation d'une implication

8.1.

a. La contraposée de $A \Rightarrow B$ est $\neg B \Rightarrow \neg A$. Une implication et sa contraposée sont-elles équivalentes ?

b. La réciproque de $A \Rightarrow B$ est $B \Rightarrow A$. Une implication et sa réciproque sont-elles équivalentes ?

c. Exprimez $\neg(A \Rightarrow B)$ avec les opérateurs $\neg$, $\wedge$ et $\vee$ mais sans parenthèses ni $\Rightarrow$.

8.2. Déterminez les contraposées, les réciproques puis les négations des propositions suivantes :

a. Si la route n'est pas droite, la pente semble forte.

b. Jamal peut franchir un cap seulement s'il change de braquet.

Exercice 9 Nouvel opérateur logique

On définit le connecteur $\oplus$ par :

$$1 \oplus 1 = 0 \quad 1 \oplus 0 = 1 \quad 0 \oplus 1 = 1 \quad 0 \oplus 0 = 0$$

9.1. Quel nom est traditionnellement donné à cet opérateur ?

9.2. Simplifier :

a. $x \oplus 0$ **b.** $x \oplus 1$ **c.** $x \oplus x$ **d.** $x \oplus \neg x$

9.3. Montrer que :

a. $x \oplus y \equiv (x \vee y) \wedge \neg (x \wedge y)$ **b.** $x \oplus y \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$

9.4. L'opération $\oplus$ est-elle commutative ?

9.5. Vrai ou faux ?

a. $x \oplus (y \oplus z) \equiv (x \oplus y) \oplus z$

b. $x \vee (y \oplus z) \equiv (x \vee y) \oplus (x \vee z)$

c. $x \oplus (y \vee z) \equiv (x \oplus y) \vee (x \oplus z)$

Exercice 10

Déterminez toutes les fonctions qui prennent deux booléens en argument et renvoient un booléen (et retrouvez le nom des fonctions classiques).

Exercice 11 jeu logique

Certains jeux font appel à des énigmes logiques régies par le calcul des propositions.

On considère la règle suivante : « *Les propositions composant une énigme sont alternativement vraies et fausses, c'est-à-dire que :*

- *soit les propositions de numéro pair sont vraies et les propositions de numéro impair fausses ;*
- *soit les propositions de numéro pair sont fausses et les propositions de numéro impair vraies.* »

Dans un labyrinthe, vous vous retrouvez bloqué dans une salle face à une porte sur laquelle se trouvent deux interrupteurs étiquetés A et B en position ouverte. On dispose de l'indication suivante :

Pour ouvrir la porte :

P1 *Il faut fermer l'interrupteur A.*

P2 *Il faut fermer simultanément les interrupteurs A et B.*

P3 *Il ne faut pas fermer l'interrupteur B.*

Attention, en cas d'erreur la salle s'auto-détruira...

11.1. Exprimer P1 , P2 et P3 sous la forme de formules du calcul des propositions dépendant de A et de B.

11.2. Exprimer la règle du jeu dans le contexte des propositions P1 , P2 et P3.

11.3. En utilisant le calcul des propositions, déterminer l'action à effectuer pour ouvrir la porte.

Exercice 12

Trois collègues, Albert, Bernard et Charles déjeunent ensemble. Les affirmations suivantes sont vraies :

1. Si Albert commande un dessert, Bernard en commande un aussi.
2. Soit Bernard, soit Charles, mais pas les deux, commandent un dessert.
3. Albert ou Charles, ou les deux, commandent un dessert.
4. Si Charles commande un dessert, Albert fait de même.

12.1.

a. Exprimer les données du problème comme des formules propositionnelles.

b. Que peut-on en déduire sur qui commande un dessert ?

c. Pouvait-on arriver à la même conclusion en supprimant l'une des quatre affirmations ?

Exercice 13

Trois personnes, A , B et C , chacune d'elle étant soit mathématicien(ne) soit informaticien(ne), ont la discussion suivante :
A : C et moi sommes mathématiciens ;
B : C n 'est pas mathématicien ;
C : B est mathématicien ou A est informaticien.

13.1. Sachant que les mathématiciens disent toujours la vérité alors que les informaticiens mentent systématiquement, pouvez-vous dire qui est quoi ?

Exercice 14

Trois autres personnes, D, E et F, chacune d'elle étant soit mathématicien(ne) soit informaticien(ne), ont la discussion suivante :
D : F est informaticien ;
E : D et F sont mathématiciens ;
F : E est mathématicien.

14.1. Sachant que les mathématiciens disent toujours la vérité alors que les informaticiens mentant systématiquement, pouvez-vous dire qui est quoi ?

Exercice 15

Dans $\mathbb{R}$, on considère les prédicats suivants :
- $p(x) : ((x < 4) \Rightarrow (x \leqslant 5))$
- $q(x) : ((x > 5) \Rightarrow (x < 2))$
- $r(x) : (x^2 \leqslant 4x)$

15.1. Quelles sont les valeurs de vérité des propositions :

a. $p(6)$
b. $q(2)$
c. $p(6) \Leftrightarrow q(2)$
d. $p(5) \wedge q(1)$
e. $r(1)$
f. $\forall x : p(x)$
g. $\exists x : p(x)$
h. $\forall x : r(x)$
i. $\exists x : q(x)$

Exercice 16

Préciser, à l'aide d'un quantificateur, le sens du mot "un" dans les phrases suivantes :

a. Youri regarde un film.
b. Un brasseur a été champion du monde de natation.
c. Un entier naturel est pair ou impair.
d. Un étudiant a toujours un sujet à étudier.
e. Dans un triangle isocèle une médiane est également hauteur.
f. Un étudiant a remarqué qu'un enseignant a toujours tendance à être mieux compris par un étudiant lorsque ce dernier est présent.

Exercice 17

On pose :

- $h(x)$: « x est un homme »
- $l(x)$: « x est un lapin »
- $g(x)$: « x est gentil »
- $m(x, y)$: « x mange y »

On considère que l'univers de référence est l'ensemble des créatures vivantes.
Traduire les phrases suivantes avec des quantificateurs :

a. « Tous les lapins sont gentils »
b. « Toutes les créatures méchantes sont des hommes »
c. « Il y a un homme qui n'est pas méchant »
d. « Il n'existe pas de lapin méchant »
e. « Aucun homme n'est gentil »
f. « Tous les hommes mangent du lapin »
g. « Un lapin a mangé un homme »

Exercice 18

Soit $\mathscr{E} = \{1, 2, 3\}$ l'univers de référence.

18.1. Déterminer la valeur de vérité des propositions suivantes :

- $\exists x, \forall y : x^2 < y + 1$
- $\exists y, \forall x : x^2 + y^2 < 12$

Exercice 19

Soit le prédicat $H(x)$ qui signifie « x est un humain », le prédicat $LG(x)$ qui signifie « x est une langue » et le prédicat $P(x, y, z)$ qui signifie « x et y parlent la langue z ».

19.1. Exprimer :

a. tous les humains parlent une langue

b. il existe une langue universelle pour les humains

c. il existe une personne qui parle toutes les langues

d. deux humains quelconques peuvent communiquer par le biais d'un interprète.

On suppose que « personne » et « interprète » sont des humains.

Exercice 20

20.1. Vrai ou faux ? (Justifier)

a. $\exists x \in \mathbb{R},\ x^2 = x$

b. $\forall x \in \mathbb{R},\ x^2 + 1 < 101$

c. $\exists x \in \mathbb{R},\ (x+1)^2 = x^2 + 2x + 1$

d. $\forall x \in \mathbb{R},\ \exists y \in \mathbb{R},\ y = x - 1$

e. $\exists y \in \mathbb{R},\ \forall x \in \mathbb{R},\ y = x - 1$

f. $\forall x \in \mathbb{R},\ \forall n \in \mathbb{N},\ x + nx = (n+1)x$

g. $\exists x \in \mathbb{R},\ \exists n \in \mathbb{N},\ x + nx = (n+1)x$

h. $\exists n \in \mathbb{N},\ \exists p \in \mathbb{N},\ n - p$ est divisible par 2

i. $\forall n \in \mathbb{N},\ \forall p \in \mathbb{N},\ n - p$ est divisible par 2

Exercice 21

Pour chaque proposition, donner sa valeur de vérité, énoncer sa négation puis donner la valeur de vérité de celle-ci :

a. $\exists x \in \mathbb{R},\ 3x = 2$

b. $\forall x \in \mathbb{R},\ x = x + 1$

c. $\forall x \in \mathbb{R},\ \forall y \in \mathbb{R},\ x \leqslant y$

d. $\exists x \in \mathbb{R},\ \forall y \in \mathbb{R},\ x^2 = y$

e. $\forall x \in \mathbb{R},\ \exists y \in \mathbb{R},\ x^2 = y$

2 Ensembles

Exercice 22 Ensembles égaux

22.1. Parmi les ensembles suivants, quels sont ceux qui sont égaux ?

a. $A = \{x \mid x \in \mathbb{R} \text{ et } x^2 - 4x + 3 = 0\}$
b. $B = \{x \mid x \in \mathbb{R} \text{ et } x^2 - 3x + 2 = 0\}$
c. $C = \{x \mid x \in \mathbb{N} \text{ et } x < 3\}$
d. $D = \{x \mid x \in \mathbb{N} \text{ et } x < 5 \text{ et } x \text{ est impair }\}$
e. $E = \{1, 2\}$
f. $F = \{1, 2, 1\}$
g. $G = \{3, 1\}$
h. $H = \{1, 1, 3\}$

Exercice 23 Ensembles définis en extension

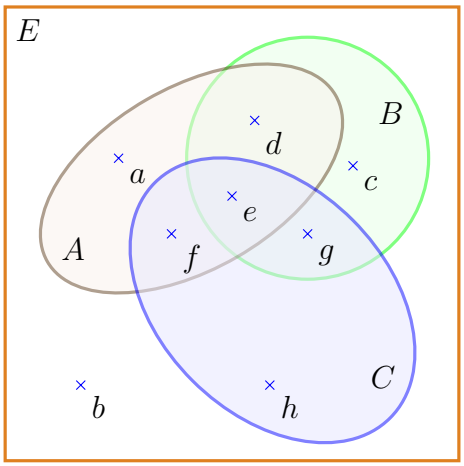
On considère l'ensemble $E = \{0, 1, 2, 3, 4, 5, 6\}$.

23.1. Définir en extension les ensembles suivants :

a. $A_1 = \{x \in \mathbb{N} \mid x^2 \in E\}$
b. $A_2 = \{x \in \mathbb{R} \mid x^2 \in E\}$
c. $A_3 = \{x \in E \mid x^2 \in E\}$
d. $A_4 = \{x \in E \mid \sqrt{x} \in E\}$
e. $A_5 = \{x \in E \mid 2x \in E\}$
f. $A_6 = \{x \in E \mid \frac{x}{2} \in E\}$

Exercice 24

On considère le diagramme de Venn suivant, avec A, B, C trois parties d'un ensemble E, et a,b,c,d,e,f,g,h des éléments de E.



24.1. Déterminez si les assertions suivantes sont vraies ou fausses :

a. $g \in A \cap \overline{B}$
b. $g \in \overline{A} \cap \overline{B}$
c. $g \in \overline{A} \cup \overline{B}$
d. $f \in C \backslash A$
e. $e \in \overline{A} \cap \overline{B} \cap \overline{C}$
f. $\{h, b\} \subset \overline{A} \cap \overline{B}$
g. $\{a, f\} \subset A \cup C$

Exercice 25 Définitions des relations ensemblistes

25.1.

a. Comment définir la relation d'inclusion $\subset$ à partir de la relation d'appartenance $\in$?

b. Comment définir la relation d'égalité ensembliste $=$ à partir de la relation d'appartenance $\in$?

c. Comment définir la relation d'égalité ensembliste $=$ à partir de la relation d'inclusion $\subset$?

Exercice 26 Propriétés des relations ensemblistes

26.1. La relation d'inclusion $\subset$ est-elle réflexive ? irréflexive ? transitive ? symétrique ? anti-symétrique ?

26.2. Même question pour la relation d'égalité ensembliste.

26.3. Qu'en est-il de la relation d'appartenance $\in$?

Exercice 27

On considère les ensembles et entiers suivants : $x \in \mathbb{N}$, $y \in \mathbb{N}$, $A \subset \mathbb{N}$, $B \subset \mathbb{N}$.

27.1. Barrer les expressions mal formées (qui n'ont pas de sens).

$A = B$	$A \in B$	$A \subset B$
$x = y$	$x \in y$	$x \subset y$
$x = A$	$x \in A$	$x \subset A$
$A = x$	$A \in x$	$A \subset x$
$A = \mathscr{P}(B)$	$A \in \mathscr{P}(B)$	$A \subset \mathscr{P}(B)$
$x = \mathscr{P}(B)$	$x \in \mathscr{P}(B)$	$x \subset \mathscr{P}(B)$
$\mathscr{P}(A) = \mathscr{P}(B)$	$\mathscr{P}(A) \in \mathscr{P}(B)$	$\mathscr{P}(A) \subset \mathscr{P}(B)$

Exercice 28

On considère l'ensemble $E = \{0, 1, 2, 3, 4\}$.

28.1. Compléter, lorsque c'est possible, par un des symboles :

$$\in, \ni, \subset, =, \neq, \not\subset, \cdots$$

1. $2 \cdots E$,
2. $\{2, 3\} \cdots E$,
3. $\{2\} \cdots E$,
4. $\{2, 3, 4\} \cdots \{4, 3, 2\}$,
5. $\{2, 3, 4\} \cdots \{4, 3, 0\}$,
6. $\{\} \cdots E$,
7. $E \cdots E$,
8. $E \cdots \{E\}$,
9. $\{\} \cdots \{E\}$,
10. $E \cdots \{0, 1, 2, 3, \ldots, 10\}$,
11. $\{0, 1, 2, 3, 4, 5\} \cdots E$

Lorsqu'il y a plusieurs solutions, on choisira celle qui donne le plus d'information.

Exercice 29

Soit $A = \{3, 5, 7\}$ et $B = \{\diamondsuit, \heartsuit\}$.

29.1. Déterminer A^2 en extension, puis B^2, $A \times B$, et $B \times A$.

29.2. Donner $A^2 \times B$ en extension.

Exercice 30 Preuve d'inclusion/égalité d'ensembles

On suppose que E et F sont deux sous-ensembles de Ω donnés en compréhension :

$$E = \{x \in \Omega \mid P(x)\} \text{ et } F = \{x \in \Omega \mid Q(x)\}$$

Lorsque l'on souhaite montrer l'inclusion $E \subset F$, une méthode classique consiste à montrer : « si $x \in E$, alors $x \in F$ ».

30.1. Déterminer une proposition logique (utilisant $P(x)$ et $Q(x)$) équivalente à cette phrase.

30.2. Déduisez-en une proposition logique à démontrer lorsque l'on souhaite montrer une égalité entre deux ensembles.

Exercice 31

Soit $A = \{(x, y) \in \mathbb{R}^2 \mid 4x - y = 1\}$ et $C = \{(t + 1, 4t + 3) \mid t \in \mathbb{R}\}$.

31.1. Démontrer que $A = C$.

Exercice 32

Soient A, B, C trois sous-ensembles de Ω.

32.1. Si $A \cup B = B \cap C$. Montrer que $A \subset B \subset C$.

32.2. Est-ce que $C \subset (A \cup B)$ entraîne $C \subset A \vee C \subset B$?

Exercice 33

On s'intéresse à la gestion de données de la prison d'aliens implantée à Roswell.

On considère un ensemble N de noms d'aliens, un ensemble de symboles $S = \{\mathsf{M}, \mathsf{F}, \mathsf{O}\}$, un ensemble P de noms de planètes et un ensemble C de numéros de cellules.

On pose $A = N \times S \times P \times C$.

On utilise un élément (n, s, p, c) de A pour associer le nom n d'un alien, son sexe s, sa planète p d'origine et le numéro c de la cellule dans laquelle il est retenu.

On dispose d'un ensemble R inclus dans A représentant l'ensemble des aliens détenus dans la prison de Roswell.

33.1. Donnez en compréhension l'ensemble des planètes d'origine des aliens détenus à Roswell.

33.2. Donnez en compréhension l'ensemble des cellules libres à Roswell.

33.3. Donnez une formule indiquant si la prison comporte deux aliens de même nom. (On fait l'hypothèse qu'il y a un seul alien par cellule).

33.4. Donnez une formule indiquant si deux aliens peuvent se reproduire dans la prison. (On fait l'hypothèse qu'il y peut y avoir plusieurs aliens par cellule).

Exercice 34 Propriétés des opérations ensemblistes

34.1. En vous appuyant sur des lois de calcul sur les connecteurs logiques $\vee$ et $\wedge$, démontrez les résultats suivants :

1. Distributivité de $\cup$ sur $\cap$.
2. Distributivité de $\cap$ sur $\cup$.
3. $\overline{A \cup B} = \overline{A} \cap \overline{B}$ *(Loi de De Morgan sur le complément et l'union)*
4. $\overline{A \cap B} = \overline{A} \cup \overline{B}$ *(Loi de De Morgan sur le complément et l'intersection)*

Exercice 35 Associativité de la différence symétrique

On cherche à comparer $(A\Delta B)\Delta C$ et $A\Delta(B\Delta C)$ avec A, B, C quelconques.

35.1. Faites 2 diagrammes de Venn sur lesquels vous colorierez successivement les ensembles $A\Delta B$ puis $(A\Delta B)\Delta C$.

35.2. Faites la même chose avec les ensembles $B\Delta C$ puis $A\Delta(B\Delta C)$.

Ces schémas ne constituent pas une preuve formelle, mais permettent de se faire une bonne intuition du résultat attendu.

35.3. Remplissez la table de vérité ci-dessous.

$a \in X$	$a \in Y$	$a \in X\Delta Y$
0	0	
0	1	
1	0	
1	1	

35.4. En déduire la table de vérité ci-dessous.

$x \in A$	$x \in B$	$x \in C$	$x \in A\Delta B$	$x \in (A\Delta B)\Delta C$	$x \in B\Delta C$	$x \in A\Delta(B\Delta C)$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

35.5. Conclure.

Exercice 36

On définit sur $\mathbb{R}$ la relation $x \mathscr{R} y$ si et seulement si $x^2 - y^2 = x - y$.

36.1. Montrer que $\mathscr{R}$ est une relation d'équivalence.

Exercice 37 Ordre lexicographique

On considère l'ensemble $\mathbb{B} = \{0, 1\}$.

37.1. Décrire une relation d'ordre sur $\mathbb{B}$.

37.2. Décrire la relation d'ordre lexicographique sur $\mathbb{B} \times \mathbb{B}$.

37.3. Cette relation est-elle bien une relation d'ordre ?

Exercice 38 Fonction injective, surjective

38.1. Déterminez si les fonctions suivantes sont injectives, surjectives, bijectives :

- $f : \begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \\ x & \mapsto & x^2+2 \end{array}$
- $g : \begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \\ x & \mapsto & x \ mod \ 3 \end{array}$
- $h : \begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \\ x & \mapsto & \begin{cases} 1 & \text{si } x \text{ est pair} \\ 0 & \text{sinon} \end{cases} \end{array}$
- $i : \begin{array}{ccc} \mathbb{N} & \to & \{0,1\} \\ x & \mapsto & \begin{cases} 1 & \text{si } x \text{ est pair} \\ 0 & \text{sinon} \end{cases} \end{array}$
- $j : \begin{array}{ccc} \mathbb{N}\times\mathbb{N} & \to & \mathbb{N} \\ (x,y) & \mapsto & x+y \end{array}$

Exercice 39 Composition de fonctions

Soit f une fonction de E vers F et g une fonction de F vers H. On note $g \circ f$ la fonction de E vers H telle que $(g \circ f)(x) = g(f(x))$.
On considère à présent les fonctions suivantes :
$f = \big\{(1,c),(2,a),(3,b),(4,c)\big\}$,
$g = \big\{(a,2),(b,1),(c,4)\big\}$ et
$h = \big\{(1,A),(2,B),(3,D),(4,D)\big\}$.

39.1. Décrivez complètement, lorsque c'est possible :

a. $g \circ f$,
b. $f \circ g$,
c. $h \circ (g \circ f)$,
d. $h \circ g$,
e. $(h \circ g) \circ f$.

Exercice 40

Soient E et F deux ensembles **finis** et f une fonction de E vers F.

40.1. Que peut-on dire des cardinaux de E et F si :

a. f est injective ?
b. f est surjective ?
c. f est bijective ?

40.2. En admettant que les conditions nécessaires soient respectées sur les cardinaux de E et F, donner le nombre de fonctions de E dans F qui soient :

a. injectives
b. surjectives
c. bijectives

Exercice 41

Soit f une fonction de A vers B et g une fonction de B vers C.

41.1. Montrer que :
- $g \circ f$ injective $\Rightarrow$ f injective,
- $g \circ f$ surjective $\Rightarrow$ g surjective.

Exercice 42

42.1. En vous appuyant sur les ensembles $\mathbb{N}$ et $\mathbb{Z}$, écrivez en compréhension l'ensemble $\mathbb{D}$ des décimaux, l'ensemble $\mathbb{Q}$ des rationnels.

42.2.

a. Donnez une bijection entre :
- $\mathbb{N}$ et $\mathbb{N}^*$,
- $\mathbb{Z}$ et $\mathbb{N}$,
- $\mathbb{N} \times \mathbb{N}$ et $\mathbb{N}$.

b. En déduire une relation entre les « cardinaux » de ces ensembles.

42.3. Donnez une surjection :
- de $\mathbb{N}$ vers $\mathbb{D}$,
- de $\mathbb{N}$ vers $\mathbb{Q}$.

Exercice 43 Plusieurs types d'infinis

Le but de cet exercice est de montrer qu'il n'existe pas de bijection entre $\mathbb{N}$ et $\mathscr{P}(\mathbb{N})$.

43.1. Déterminer une injection de $\mathbb{N}$ dans $\mathscr{P}(\mathbb{N})$.

43.2. On suppose l'existence d'une bijection f entre $\mathbb{N}$ et $\mathscr{P}(\mathbb{N})$.

a. si $x \in \mathbb{N}$, quel est le type de $f(x)$?

b. Soit $A = \{x \in \mathbb{N} \mid x \notin f(x)\}$. Montrer que $A \in \mathscr{P}(\mathbb{N})$.

c. En déduire qu'il existe un $y \in \mathbb{N}$ tel que $f(y) = A$.

d. Est-ce que $y \in A$? Est-ce que $y \notin A$?

e. Conclure.

*Ce résultat montre que $\mathscr{P}(\mathbb{N})$ contient strictement plus d'éléments que $\mathbb{N}$. Il existe donc une différence entre l'infini du cardinal de $\mathbb{N}$ et l'infini du cardinal de $\mathscr{P}(\mathbb{N})$. En mathématique, on dit qu'un ensemble en bijection avec $\mathbb{N}$ est **dénombrable**, $\mathscr{P}(\mathbb{N})$ est donc un ensemble **indénombrable**.*

Travaux Pratiques

3 Découverte de Python

Exercice 1

L'objectif de ce TP est de vous familiariser avec Python.

Python peut être lancé de deux manières :

- via la console : tapez `python3` dans un terminal
- en exécutant un fichier : tapez `python3 nom_fichier.py` dans un terminal

1.1. Tapez les commandes suivantes :

```
>>> 5+7
```

```
>>> 34/9
```

```
>>> min(3,5)
```

```
>>> sqrt(9)
```

Les fonctions mathématiques ne sont pas chargées par défaut : il faut importer le module `math` :

1.2. Tapez les commandes suivantes :

```
>>> import math
```

```
>>> math.sqrt(9)
```

Pour utiliser la fonction `sqrt` du package `math`, il y a 3 possibilités :

- importer le package `math` puis utiliser la commande `math.sqrt`
- importer le package `math` en lui donnant un alias :
  ```
  >>> import math as mt
  ```
  puis utiliser la commande `mt.sqrt`. Ceci permet d'éviter des noms trop long de package.
- importer directement certaines ou toutes les fonctions du package `math` :
  ```
  >>> from math import *
  ```
  puis les utiliser directement sans préfixe : `sqrt`. ATTENTION !! : ceci est risqué si vous importez plusieurs packages avec des fonctions portant le même nom.

1.3. Calculer les valeurs suivantes :

- $\sqrt{5}$
- 2^6
- $17 \ mod \ 4$

Les variables s'affectent par le signe `=` :

```
>>> a = 3+4
```

L'affichage d'une variable dans la console se fait juste en tapant son nom ou `print(nom_variable)`

```
>>> a
```

```
>>> print(a)
```

En Python, les deux valeurs binaires sont `True` et `False`.

1.4. (Opérateurs binaires) Exécutez les commandes suivantes :

- `True and False`
- `True or False`
- `not(True) or False and True`

==	teste l'égalité entre deux valeurs
!=	teste la différence entre deux valeurs
<, <=, >, >=	teste les relations d'ordre entre deux valeurs

Comparaisons

Exercice 2 `if`, `while`, `for`, `range` et fonctions

Voici un exemple d'utilisation en python de l'instruction `if` :

```
if x>0 :
        y=x
        print("Cas positif")
elif x<0 :
        y=-x
        print("Cas négatif")
else :
        y=0
        print("Cas nul")
print("Fin")
```

Noter l'absence d'accolades et l'utilisation de l'indentation : les blocs de l'instruction conditionnelle ne sont distinguables que par l'indentation. Il en est de même pour les boucles `for`, `while` et pour l'écriture des fonctions.

Voici un exemple de fonction écrite en python :

```
def f(x) :
        if x >= 0 :
                return x
        else :
                return -x
```

2.1. Dans un fichier `tutoriel.py`, écrivez une fonction g permettant de calculer :

$$g(x) = \begin{cases} \sqrt{-x} & \text{si } x < 0 \\ \sqrt{x} & \text{sinon} \end{cases}$$

Python est surtout un langage de script. Si vous souhaitez tester votre fonction, il vous suffit de rajouter, par exemple,

```
print(g(-4))
```

à la fin de votre fichier et de le lancer dans le terminal par la commande :

```
$ python3 tutoriel.py
```

La commande `for` s'utilise de la manière suivante :

```
for i in [4,8,-3] :
        print(i)
```

2.2. Afficher ce que donne les commandes :

- ```
  for i in range(6) :
      print(i)
  ```
- ```
  for i in range(5,10) :
      print(i)
  ```
- ```
  for i in range(30,3,-7) :
      print(i)
  ```

2.3. Écrivez une fonction h prenant en paramètre un entier positif x et affichant tous les entiers multiples de 7 compris entre 0 et x.

Exercice 3 Tuples et listes

En python, une liste s'écrit entre crochets, avec ses valeurs séparées par des virgules :

```
>>> squares = [1, 4, 9, 16, 25]
```

3.1. Déterminer ce qu'affichent et modifient les commandes suivantes :

- squares[0]
- squares[1:3]
- squares[:2]
- squares[2:]
- squares[−1]
- squares+36
- squares∗2
- squares∗3.5
- squares+[36]
- squares.append(36)
- squares.append([49])
- squares.extend([64])
- squares.remove(25)
- squares.pop(4)

3.2. Construire une fonction `compte` prenant en paramètres un élément `el` et une liste `liste` et retournant le nombre d'éléments de `liste` égaux à `el`.

3.3. Construire une fonction `carres` prenant en paramètre une liste `liste` et retournant une liste contenant les carrés des éléments de `liste`.

Les tuples (ou n-uplets) sont une deuxième structure classique en python, voici un exemple de déclaration :

```
>>> mon_tuple = (1,8,8,64,125)
```
```

Les sets (ou ensembles) sont une troisième structure en python, voici un exemple de déclaration (à remarquer notamment la façon dont est géré le 8 qui apparaît en doublon) :

>>>mon_set={1,8,8,64,125}

3.4. Déterminer quelles actions possibles sur les listes ne le sont pas sur les tuples et/ou sur les ensembles.

3.1 Constructions de base

Python est muni d'un mécanisme de typage dynamique. Le type d'une variable est donc défini selon la valeur qui lui est affectée et peut changer en cas de nouvelle affectation. Il n'est donc pas nécessaire de déclarer un type avec la variable. On peut interroger le type d'un variable en utilisant la fonction `type(var)` ou préférentiellement (si la variable est susceptible d'être un "sous-type" du type testé) en questionnant avec la fonction `isinstance(var,type)`. Python dispose par ailleurs de la valeur particulière `None` qui peut être vue comme une constante utile pour exprimer l'absence de valeur (par exemple pour une fonction qui ne renvoie pas toujours de résultat).
Les types de base sont :

- `int` : entiers relatifs sur lesquels portent les opérateurs de calculs habituels (dont la puissance `**`) et aussi en particulier les opérateurs de quotient `//` et de reste `%` liés à la division euclidienne.
- `float` : nombre en virgule flottante
- `bool` : booléens `True` et `False`
- `str` : chaînes de caractères encadrées par des apostrophes ou guillemets, et sur lesquelles s'applique l'opérateur `+` de concaténation

Les objets créés avec ces types sont immuables. Cela signifie qu'après avoir créé un objet de ce type et lui assigné une valeur, on ne peut plus modifier cette valeur. Cela n'interdit pas pour autant d'affecter un nouvel objet de ce type à une variable existante.

```
>>>mot='toto'
>>>mot[1]
'o'
>>>mot[1]='a'
TypeError: 'str' object does not support item assignment
>>>mot='tata'
>>>mot
'tata'
```

Différentes structures de données

- `list` : une liste est une structure indexée et mutable. Doublons autorisés.
- `tuple` : un n-uplet est une structure indexée et immuable. Doublons autorisés.
- `set` : un ensemble est une structure non indexée et mutable. Pas de doublon autorisé.
- `dict` : un dictionnaire est une structure où l'indexation peut se faire par une chaîne de caractères. Chaque élément d'un dictionnaire s'apparente à un couple

(clé,valeur) où la clé est la chaîne de caractères désignant l'indexation de la valeur dans le dictionnaire. Le dictionnaire est mutable. Pas de doublon autorisé.
- `array` : les tableaux possèdent les mêmes propriétés de base que les listes mais ils nécessitent l'import d'une bibliothèque pour être utilisés (`array` ou `numpy`)

Nous allons ci-après détailler les manipulations de listes. Un certain nombre de ces manipulations seront aussi valables sur les autres structures. Leurs propriétés respectives notamment en termes d'indexation, de mutabilité rendront néanmoins certaines opérations possibles ou non et des précautions devront être prises dans la copie ou l'appel à une variable à différents emplacements d'un programme selon la propriété de mutabilité.

3.2 Les listes

Quelques fonctionnalités pratiques

Définition d'une liste en extension Une liste peut-être définie en extension en faisant figurer entre crochets ses éléments séparés par des virgules, et la liste vide s'écrit [].

Longueur d'une liste La fonction `len` permet d'obtenir la longueur d'une liste.

```
>>>len([0,1,2])
3
```

Accès aux éléments d'une liste Lorsque `l` est une liste, l'expression `l[i]` désigne le `i`-ème élément de cette liste (avec $i > 0$). La position du premier élément d'une liste étant l'entier 0, une liste contenant n éléments permettra l'accès à un élément dont la position sera définie entre 0 et $n-1$. On peut également accéder aux éléments de la liste en indexant avec des nombres négatifs à partir de la droite (`l[-1]` désigne le dernier élément, `l[-2]` l'avant dernier...)

Dans le cas par exemple où une fonction renvoie une liste x, il peut parfois être pratique de déconstruire cette liste avec une instruction d'affectation du type

$$x_0, x_1, \ldots, x_{n-1} = x$$

qui permet d'affecter la valeur de $x[i]$ à chaque variable x_i.

Listes et affectations Lorsque `l1` est un nom de variable désignant une liste, la valeur associée au nom `l1` est la référence à la zone mémoire où sont stockés les éléments de la liste. Aussi, à l'issue de l'affectation `l2=l1`, du fait de la propriété de mutabilité des listes, la nouvelle valeur de `l2` est la référence à la zone mémoire où sont stockés les éléments de la liste `l1`, et si on modifie `l1` (resp. `l2`), alors on modifie également `l2` (resp. `l1`).

```
>>>l1=[4,7,1]
>>>l2=l1
>>>l1[1]=0
>>>l2
[4,0,1]
```

La liste `l1[:]` désigne une copie de tous les éléments de la liste `l1` et est stockée dans une zone mémoire qui lui est propre (on peut aussi créer une copie en utilisant

la méthode `l1.copy()`). Une liste `l3` ainsi créée comme copie de `l1` ne sera donc pas modifiée en cas de modifications apportées sur `l1` (et vice-versa).

Listes en paramètres de fonction Puisque la valeur d'un nom désignant une liste correspond à une référence à la zone mémoire où sont stockés les éléments de cette liste, lorsqu'une fonction est appelée avec ce nom pour argument, c'est cette référence qui est transmise et utilisée dans le corps de la fonction. Aussi si le corps de cette fonction contient une instruction qui modifie la valeur d'un élément de cette liste, c'est la liste utilisée pour appeler la fonction qui est modifiée. L'exemple suivant illustre ce mécanisme :

```
def double_list(l):
      for i in range(0,len(l)):
            l[i]=2*l[i]
      return l
>>>l=[1,2,3,4]
>>>double_list(l)
[2,4,6,8]
>>>l
[2,4,6,8]
```

Pour ne pas modifier la liste utilisée lors de l'appel à cette fonction, il suffit d'appeler cette fonction avec une copie de la liste :

```
>>>l=[1,2,3,4]
>>>double_list(l[:])
[2,4,6,8]
>>>l
[1,2,3,4]
```

Sous-liste d'une liste Lorsque `l` est une liste, l'expression `l[i:j]` désigne la liste des éléments de `l` situés entre les positions de `i` et `j-1` dans `l` (si $j > i$, sinon c'est la liste vide). L'expression `l[:j]` est équivalente à l'expression `l[0:j]`, l'expression `l[i:]` est équivalente à l'expression `l[i:len(l)]` et l'expression `l[-k:]` avec $k > 0$ renvoie la liste des `k` derniers éléments de la liste `l`

Intervalle d'entiers Python permet de manipuler des intervalles d'entiers représentés par des listes d'entiers : `range(n,m)` désigne la liste $[n, n+1, \ldots, m-1]$ (lorsque $m > n$). On peut aussi utiliser la construction `range` avec un unique paramètre entier : `range(n)` désigne l'intervalle $[0, \ldots, n-1]$.

Itération sur les éléments d'une liste La construction `for` permet de répéter l'exécution d'un bloc d'instructions, délimité par l'indentation, en parcourant les éléments d'une liste. Par exemple, la fonction suivante permet de calculer la somme des éléments d'une liste de nombres.

```
def somme(l) :
      r=0
      for e in l:
            r=r+e
      return r
```

Définition en compréhension Python permet de définir une liste en compréhension. La construction `[e for x in s]`, où `x` est une variable de compréhension ou un n-uplet de variables, `s` est une séquence (une liste par exemple) et `e` est une expression (qui porte sur `x`), permet de construire la liste des valeurs successives de `e` obtenues à partir des valeurs successives de `x` issues du parcours de la séquence `s`.

```
>>>[x**2 for x in range(2,5)]
[4,9,16]
>>>[x+y for x,y in [(1,2),(3,4),(5,6)]
[3,7,11]
```

3.3 Les ensembles

Initialiser un ensemble On peut initialiser un ensemble vide en utilisant `set()`. Pour initialiser un ensemble avec des valeurs, on place ces valeurs entre deux accolades (l'ordre n'a pas d'importance puisque l'ensemble n'est pas ordonné). Attention la commande {} déclare un dictionnaire vide et non un ensemble.

Ajouter ou retirer des valeurs On peut utiliser la méthode `add` pour ajouter une valeur (`append` définie pour une liste n'est pas définie pour un ensemble) et la méthode `remove` pour en retirer une.

```
>>>radios_lucien=set()
>>>radios_lucien.add('nrj')
>>>radios_lucien
{'nrj'}
>>>radios_walid={'france info','france inter','fip' }
>>>radios_walid.remove('france info')
>>>radios_walid
{'fip','france inter'}
```

Remarque : L'ensemble renvoyé est présenté avec un ordre alphabétique par défaut

On ne peut ajouter que des valeurs mutables à un ensemble. Un `TypeError` apparaîtra si vous tentez d'ajouter une liste à un ensemble. A noter que si vous ajoutez un tuple, la valeur ajoutée aura le statut de tuple. Pour ajouter un ensemble de valeurs, on peut faire la réunion de l'ensemble initial avec l'ensemble contenant les valeurs à ajouter.

Des méthodes pour des opérations ensemblistes

On dispose pour les opérations ensemblistes des méthodes suivantes : `union`, `intersection`, `difference`, `symmetric_difference`

Quelques tests

- `A.isdisjoint(B)` Méthode pour interroger si les ensembles `A` et `B` sont disjoints
- 'france info' in radios_walid pour tester l'appartenance
- A.issubset(B) pour tester l'inclusion de A dans B.

3.4 La portée des variables

En Python, nous pouvons déclarer des variables n'importe où dans notre script : au début du script, à l'intérieur des boucles, au sein de nos fonctions, etc. L'endroit où

on définit une variable va néanmmoins déterminer l'endroit où la variable va être accessible. L'expression "portée des variables" sert à désigner les différents espaces dans le script dans lesquels une variable est accessible. En Python, une variable peut avoir une portée locale ou globale et on peut lister les variables enregistrées dans ces espaces de noms en utilisant les fonctions `globals()` et `locals()`.

Les variables définies dans une fonction sont appelées variables locales. Elles ne peuvent être utilisées que localement c'est-à-dire qu'à l'intérieur de la fonction qui les a définies. Appeler une variable locale depuis l'extérieur de la fonction qui l'a définie provoquera une erreur.

```
def ajoute(x):
      n=2
      l=[2]
      n=n+x
      l.append(x)
      print('n=',n)
      print('l=',l)
>>>ajoute(3)
n= 5
l= [2, 3]
>>>print(n)
NameError: name 'n' is not defined
>>>print(l)
NameError: name 'l' is not defined
```

Une variable définie au début du script sera enregistrée comme globale. Les fonctions exploiteront les variables définies localement, mais en dehors des fonctions ce seront les noms associés aux variables définies globalement qui seront exploités

```
n=1
l=[1]
def ajoute(x):
      n=2
      l=[2]
      n=n+x
      l.append(x)
      print('n=',n)
      print('l=',l)
>>>ajoute(3)
n= 5
l= [2, 3]
>>>print(n)
1
>>>print(l)
[1]
```

Si une variable est appelée dans une fonction sans qu'une variable n'y a été déclarée, le programme exploitera une variable globale portant ce nom uniquement si celle-ci est mutable et auquel cas l'exécution de la fonction pourra modifier la valeur de cette variable.

Si une variable globale portant ce même nom existe mais est immuable, l'exécution

de la fonction renverra une erreur.

```
l=[1]
def ajoute(x):
      l.append(x)
      print('l=',l)
>>>ajoute(3)
l= [1, 3]
>>>print(l)
[1, 3]
```

```
n=1
def ajoute(x):
      n=n+x
      print('n=',n)
>>>ajoute(3)
local variable 'n' referenced before assignment
```

Dans le cas d'une variable immuable pour forcer la fonction à l'utiliser quitte à y assigner une nouvelle valeur on devra déclarer dans la fonction l'appel à cette variable avec le mot clé `global`

```
n=1
def ajoute(x):
      global n
      n=n+x
      print('n=',n)
>>>ajoute(3)
n= 4
>>>print(n)
4
```

4 Logique

Exercice 4

4.1. En utilisant seulement des conditionnelles (`if`) et sans les opérateurs logiques de `python` (`and`, `or`, `not`), coder les opérateurs logiques suivants :

- `et(x,y)` qui renvoie `x` $\wedge$ `y`
- `ou(x,y)` qui renvoie `x` $\vee$ `y`
- `non(x)` qui renvoie $\neg$`x`

5 Ensembles

Exercice 5 Parcours d'ensembles en `python`

Dans la question 5.1, on utilise des `set` pour représenter des ensembles.
Le but de ces TP est de vous faire manipuler les concepts de base de programmation, aussi il est important que vous codiez les fonctions sans utiliser les fonctions prévues dans `python` pour manipuler les `set`.

5.1. Écrire en `python` les fonctions suivantes :

a. `affiche_col(E)` : fonction qui affiche un par un les éléments de l'ensemble `E` passé en argument.

b. `membre(x,E)` : fonction qui prend un élément `x` et un ensemble `E` en paramètres et qui renvoie `True` si `x` est dans `E`, `Faux` sinon.

c. `somme(E)` : fonction qui calcule, quand c'est possible, la somme des éléments d'un ensemble `E`;

d. `taille(E)` : calcule le nombre d'éléments d'un ensemble `E`;

e. `moyenne(E)` : calcule, quand c'est possible, la moyenne des éléments d'un ensemble `E`;

5.2. Un ensemble est un regroupement d'éléments sans doublon et sans ordre :

a. écrivez une fonction `est_ensemble(l)` prenant en paramètre une liste l et qui retourne `True` si la liste `l` ne possède pas de doublon, `False` sinon.

b. écrivez une fonction `suppr_doublon(l)` qui retourne l'ensemble issu de la liste `l`, c'est à dire sans doublon.

Exercice 6 Opérations ensemblistes en `python`

6.1. Écrire en `python` les fonctions booléennes suivantes :

a. `inclus(A,B)` : renvoie `True` si l'ensemble représenté par `A` est inclus dans celui représenté par `B`, `False` sinon.

b. `egal(A,B)` : renvoie `True` si l'ensemble `A` est égal à l'ensemble `B`, `False` sinon.

c. `disjoint(A,B)` : renvoie `True` si les ensembles `A` et `B` sont disjoints, `False` sinon.

6.2. Écrire en `python` les opérations ensemblistes suivantes :

a. `ajout(x,E)` : retourne le résultat de l'ajout de l'élément `x` à l'ensemble `E` ;

b. `union(A,B)` : retourne l'union de `A` et `B` ;

c. `intersection(A,B)` : retourne l'intersection de `A` et `B` ;

d. `retire(x,E)` : retourne le résultat de la suppression de l'élément `x` dans l'ensemble `E` ;

e. `diff(A,B)` : retourne l'ensemble $A \setminus B$

f. `diff_sym(A,B)` : retourne l'ensemble $A \Delta B$

6 Arithmétique

Exercice 7 Algorithmes abordés dans le cours

7.1. Réaliser les programmes suivants sans faire appel aux opérateurs ou fonctions déjà intégrés dans Python permettant le renvoi de ces résultats.

a. Programmer l'algorithme prenant en arguments deux entiers naturels a et b et renvoyant le couple formé du quotient et du reste de la division euclidienne de a par b (sans faire appel aux opérateurs déjà intégrés dans Python pour le renvoi de ces résultats)

b. Programmer l'algorithme d'Euclide étendu prenant en arguments deux entiers naturels a et b et renvoyant le triplet formé du pgcd d de ces deux entiers et de deux entiers u et v vérifiant $au + bv = d$.

c. Programmer l'algorithme prenant en arguments deux k-uplets de valeurs $(a_1, a_2, \cdots, a_k)$ et $(n_1, n_2, \cdots, n_k)$ et renvoyant, si les (n_i) sont deux à deux premiers entre eux, le couple d'entiers (x, n) tel que $0 \leq x \leq n = \prod_{i=1}^{k} n_i$ et

$$\begin{cases} x \equiv a_1[n_1] \\ \cdots \\ x \equiv a_k[n_k] \end{cases}$$

7.2. Je disposais intialement d'un sachet de trente billes et globalement j'en ai plutôt perdu alors que normalement je suis plutôt bon, je comprends pas. Bref toujours est-il que si je les dispose :

- en tas de 2 billes il m'en reste une toute seule
- en tas de trois billes, il n'en reste pas
- en tas de 5 billes, il en reste deux

Combien me reste-t-il de billes ?

Exercice 8 Test de primalité

8.1. Justifier qu'un nombre p est premier si et seulement si il n'est divisible par aucun nombre premier inférieur ou égal à $\sqrt{p}$

8.2. Ecrire une fonction renvoyant le $n^{\text{ème}}$ nombre premier en exploitant au mieux l'idée précédente.

Exercice 9 Conversion bases système numération

9.1. Ecrire une fonction permettant la conversion d'un nombre n en écriture décimale en nombre en base $b < 10$ soit un n-uplet de valeurs dans $\{0, \ldots, b-1\}$.

9.2. Ecrire une fonction inverse de la première

Exercice 10 Calcul de $a^k(\bmod n)$

Dans l'exemple ci-dessous on exploite le développement de k en base 2 pour le calcul de $5^{11}(\bmod 14)$. Ecrire un programme vous inspirant de ceci. Pouvez-vous remarquer un gain en temps d'exécution par rapport à un calcul direct ?

- Conversion de l'exposant en base 2 :
  $11 = 8 + 2 + 1 = 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0$
- Calcul des $5^{2^i}(\bmod 14)$:
  $5^{2^0} \equiv 5(\bmod 14)$
  $5^{2^1} \equiv 5 \times 5 \equiv 25 \equiv 11(mod 14)$
  $5^{2^2} \equiv 5^{2^1} \times 5^{2^1} \equiv 11 \times 11 \equiv 121 \equiv 9(\bmod 14)$
  $5^{2^3} \equiv 5^{2^2} \times 5^{2^2} \equiv 9 \times 9 \equiv 81 \equiv 11(\bmod 14)$
- Conclusion :

$$\begin{aligned} 5^{11}(\bmod 14) &\equiv 5^{2^3} \times 5^{2^1} \times 5^{2^0} \\ &\equiv 11 \times 11 \times 5(\bmod 14) \\ &\equiv 605(\bmod 14) \\ &\equiv 3(\bmod 14) \end{aligned}$$

Présentation du principe du codage RSA

Principe général

Le chiffrement RSA est asymétrique : il utilise une paire de clés (des nombres entiers) composée d'une clé publique pour chiffrer et d'une clé privée pour déchiffrer des données confidentielles. Les deux clés sont créées par une personne, souvent nommée par convention Alice, qui souhaite que lui soient envoyées des données confidentielles (un message envoyé sera sous la forme d'un nombre entier). Alice rend la clé publique accessible. Cette clé est utilisée par ses correspondants (Bob, etc.) pour chiffrer les données qui lui sont envoyées. La clé privée est quant à elle réservée à Alice, et lui permet de déchiffrer ces données. Le fonctionnement du cryptosystème RSA est basé sur la difficulté d'exprimer de grands entiers comme produits d'entiers.

Création des clés L'étape de création des clés est à la charge d'Alice.

- Elle choisit deux nombres premiers p et q et calcule le module de chiffrement $n = pq$ (les entiers envoyés par messages devront être inférieurs à n (à noter que la sécurité du cryptage repose sur le fait que p et q soient grands)
- Elle calcule $\varphi(n) = (p-1)(q-1)$, soit la valeur de l'indicatrice d'Euler en n (égalité à démontrer en partant de la définition qui suit : la fonction indicatrice d'Euler associe à tout entier naturel n non nul le cardinal, noté $\varphi(n)$, de l'ensemble des nombres naturels inférieurs à n et premiers avec n)
- Elle cherche un nombre entier e tel que $e \wedge \varphi(n) = 1$. e est appelé exposant de chiffrement et le couple (n, e) constitue la clé publique d'Alice.
- Elle cherche ensuite l'unique nombre entier d inverse de e modulo $\varphi(n)$ et strictement inférieur à $\varphi(n)$ (l'existence de d tient au fait que $e \wedge \varphi(n) = 1$ et on peut le déterminer grâce à l'algorithme d'Euclide étendu pour obtenir l'identité de Bézout (Alice a besoin pour l'obtenir de $\varphi(n) = (p-1)(q-1)$ et donc de p et q qui ne sont connus que d'elle). Le nombre d, appelé exposant de déchiffrement, constitue la clé privée.

Chiffrement du message On note M le message que souhaite envoyer Bob. M est un entier naturel strictement inférieur à n. Le message chiffré sera représenté par l'entier naturel C strictement inférieur à n et tel que :

$$M^e \equiv C[n]$$

Déchiffrement du message Pour déchiffrer C, on utilise d, car on peut montrer que :

$$M \equiv C^d[n]$$

Justification On doit montrer que $M^{ed} \equiv M[n]$.
d étant l'inverse de e modulo $\varphi(n)$, on en déduit qu'il existe un nombre entier k tel que $ed = 1 + k\varphi(n)$ ou encore $ed = 1 + k(p-1)(q-1)$. On a alors :

$$M^{ed} \equiv M^{1+k(p-1)(q-1)}$$

On envisage alors deux premiers cas :

- Si M n'est pas un multiple de p alors on a d'après le petit théorème de Fermat : $M^{p-1} \equiv 1[p]$. En exprimant alors $M^{1+k(p-1)(q-1)}$ comme $M \times (M^{p-1})^{k(q-1)}$, on obtient que $M^{ed} \equiv M[p]$
- De manière identique si M n'est pas un multiple de q alors on a : $M^{q-1} \equiv 1[q]$. Et en exprimant alors $M^{1+k(p-1)(q-1)}$ comme $M \times (M^{q-1})^{k(p-1)}$, on obtient que $M^{ed} \equiv M[q]$

Ces deux égalités sont en fait vérifiées pour tout entier M car si M est multiple de p, M et toutes ses puissances avec un exposant non nul sont congrues à 0 modulo p (donc $M^{ed} \equiv M[p]$). Et on obtient de même que si M est multiple de q on retrouve également que $M^{ed} \equiv M[q]$. L'entier $M^{ed} - M$ est donc un multiple de p et de q, qui sont des nombres premiers distincts donc premiers entre eux. $M^{ed} - M$ est donc également multiple de leur produit $pq = n$ (conséquence du théorème de Gauss). On a donc bien démontré que $M^{ed} \equiv M[n]$.