COURS PI

L'école sur-mesure

de la Maternelle au Bac, Établissement d'enseignement privé à distance, déclaré auprès du Rectorat de Paris

Première - Module 2 - Traitement des données

Numérique et Sciences Informatiques

v.5.1

- ***Guide de méthodologie***
  pour appréhender notre pédagogie
- ***Leçons détaillées***
  pour apprendre les notions en jeu
- ***Exemples et illustrations***
  pour comprendre par soi-même
- ***Prolongement numérique***
  pour être acteur et aller + loin
- ***Exercices d'application***
  pour s'entraîner encore et encore
- ***Corrigés des exercices***
  pour vérifier ses acquis

www.cours-pi.com

Paris & Montpellier

EN ROUTE VERS LE BACCALAURÉAT

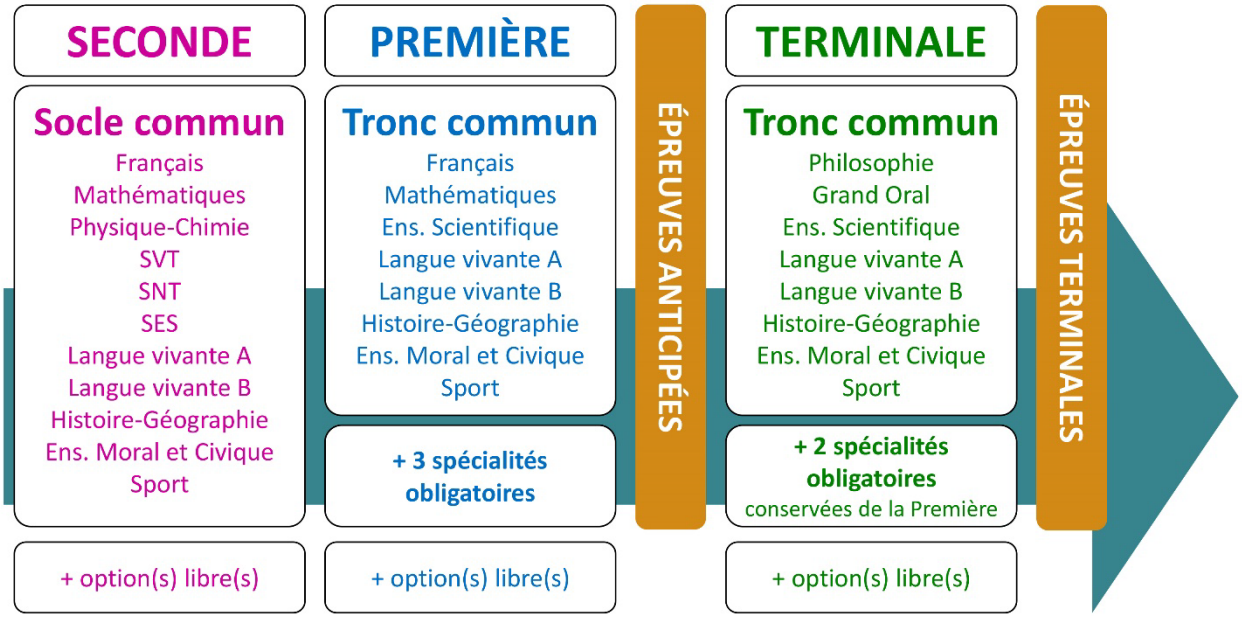
Comme vous le savez, **la réforme du Baccalauréat** est entrée en vigueur progressivement jusqu'à l'année 2021, date de délivrance des premiers diplômes de la nouvelle formule.

Dans le cadre de ce nouveau Baccalauréat, **notre Etablissement**, toujours attentif aux conséquences des réformes pour les élèves, s'est emparé de la question avec force **énergie** et **conviction** pendant plusieurs mois, animé par le souci constant de la réussite de nos lycéens dans leurs apprentissages d'une part, et par la **pérennité** de leur parcours d'autre part. Notre Etablissement a questionné la réforme, mobilisé l'ensemble de son atelier pédagogique, et déployé tout **son savoir-faire** afin de vous proposer un enseignement tourné continuellement vers l'**excellence**, ainsi qu'une scolarité tournée vers la **réussite**.

- Les **Cours Pi** s'engagent pour faire du parcours de chacun de ses élèves un **tremplin vers l'avenir**.
- Les **Cours Pi** s'engagent pour ne pas faire de ce nouveau Bac un diplôme au rabais.
- Les **Cours Pi** vous offrent **écoute** et **conseil** pour coconstruire une **scolarité sur-mesure**.

LE BAC DANS LES GRANDES LIGNES

Ce nouveau Lycée, c'est un enseignement à la carte organisé à partir d'un large tronc commun en classe de Seconde et évoluant vers un parcours des plus spécialisés année après année.



CE QUI A CHANGÉ

- Il n'y a plus de séries à proprement parler.
- Les élèves choisissent des spécialités : trois disciplines en classe de Première ; puis n'en conservent que deux en Terminale.
- Une nouvelle épreuve en fin de Terminale : le Grand Oral.
- Pour les lycéens en présentiel l'examen est un mix de contrôle continu et d'examen final laissant envisager un diplôme à plusieurs vitesses.
- Pour nos élèves, qui passeront les épreuves sur table, le Baccalauréat conserve sa valeur.

CE QUI N'A PAS CHANGÉ

- Le Bac reste un examen accessible aux candidats libres avec examen final.
- Le système actuel de mentions est maintenu.
- Les épreuves anticipées de français, écrit et oral, tout comme celle de spécialité abandonnée se dérouleront comme aujourd'hui en fin de Première.

A l'occasion de la réforme du Lycée, nos manuels ont été retravaillés dans notre atelier pédagogique pour un accompagnement optimal à la compréhension. Sur la base des programmes officiels, nous avons choisi de créer de nombreuses rubriques :

- **À vous de jouer** pour mettre en pratique le raisonnement vu dans le cours et s'accaparer les ressorts de l'analyse, de la logique, de l'argumentation, et de la justification
- **Pour aller plus loin** pour visionner des sites ou des documentaires ludiques de qualité
- Et enfin ... la rubrique **Les Clés du Bac by Cours Pi** qui vise à vous donner, et ce dès la seconde, toutes les cartes pour réussir votre examen : notions essentielles, méthodologie pas à pas, exercices types et fiches étape de résolution !

NUMÉRIQUE ET SCIENCES INFORMATIQUES PREMIÈRE

Module 2 – Traitement des données

L'AUTEUR

Adrien SAURAT

« L'enseignement, c'est favoriser l'autonomie et l'enrichissement des élèves, avec en autres objectifs, apprendre un métier. » Professeur et formateur en informatique avec plus de douze ans d'expérience en développement web et dans l'animation du réseau Canopé, il se passionne aussi pour le théâtre et l'écriture de nouvelles.
Des passions qui l'ont déjà conduit sur les planches du Festival d'Avignon.

PRÉSENTATION

Ce **cours** est divisé en chapitres, chacun comprenant :

- Le **cours**, conforme aux programmes de l'Education Nationale
- Des **applications** dont les **corrigés** se trouvent en **fin de chapitre**
- Des **exercices d'entraînement** et leurs **corrigés** en **fin de fascicule**
- Des **devoirs** soumis à correction (et ***se trouvant hors manuel***). Votre professeur vous renverra le corrigé-type de chaque devoir après correction de ce dernier.

Pour une manipulation plus facile, les corrigés-types des exercices d'application et d'entraînement sont regroupés en fin de manuel.

CONSEILS A L'ÉLÈVE

Vous disposez d'un support de cours complet : **prenez le temps** de bien le lire, de le comprendre mais surtout de l'**assimiler**. Vous disposez pour cela d'exemples donnés dans le cours et d'exercices types corrigés.
Vous pouvez rester un peu plus longtemps sur une unité mais travaillez régulièrement.

LES DEVOIRS

Les devoirs constituent le moyen d'évaluer l'acquisition de vos *savoirs* (« Ai-je assimilé les notions correspondantes ? ») et de vos *savoir-faire* (« Est-ce que je sais expliquer, justifier, conclure ? »).

Placés à des endroits clés des apprentissages, ils permettent la vérification de la bonne assimilation des enseignements.

Aux *Cours Pi*, vous serez accompagnés par un professeur selon chaque matière tout au long de votre année d'étude. Référez-vous à votre « Carnet de Route » pour l'identifier et découvrir son parcours.
Avant de vous lancer dans un devoir, assurez-vous d'avoir **bien compris les consignes**.
Si vous repérez des difficultés lors de sa réalisation, n'hésitez pas à le mettre de côté et à revenir sur les leçons posant problème. **Le devoir n'est pas un examen**, il a pour objectif de s'assurer que, même quelques jours ou semaines après son étude, une notion est toujours comprise.
Aux *Cours Pi*, chaque élève travaille à son rythme, parce que chaque élève est différent et que ce mode d'enseignement permet le « sur-mesure ».
Nous vous engageons à respecter le moment indiqué pour faire les devoirs. Vous les identifierez par le bandeau suivant :

Vous pouvez maintenant
faire et envoyer le devoir n°1

Il est **important de tenir compte des remarques, appréciations et conseils du professeur-correcteur**. Pour cela, il est **très important d'envoyer les devoirs au fur et à mesure** et non groupés. **C'est ainsi que vous progresserez !**

Donc, dès qu'un devoir est rédigé, envoyez-le aux *Cours Pi* par le biais que vous avez choisi :

1) Par **soumission en ligne** via votre espace personnel sur **PoulPi**, pour un envoi **gratuit**, **sécurisé** et plus **rapide**.
2) Par **voie postale** à *Cours Pi*, 9 rue Rebuffy, 34 000 Montpellier
   *Vous prendrez alors soin de joindre une **grande enveloppe libellée à vos nom et adresse**, et **affranchie** **au tarif en vigueur** pour qu'il vous soit retourné par votre professeur*

***N.B. :** quel que soit le mode d'envoi choisi, vous veillerez à **toujours joindre l'énoncé du devoir ;** plusieurs énoncés étant disponibles pour le même devoir.*
***N.B. :** si vous avez opté pour un envoi par voie postale et que vous avez à disposition un scanner, nous vous engageons à conserver une copie numérique du devoir envoyé. Les pertes de courrier par la Poste française sont très rares, mais sont toujours source de grand mécontentement pour l'élève voulant constater les fruits de son travail.*

SOUTIEN ET DISPONIBILITÉ

VOTRE RESPONSABLE PÉDAGOGIQUE

Professeur des écoles, professeur de français, professeur de maths, professeur de langues : notre Direction Pédagogique est constituée de spécialistes capables de dissiper toute incompréhension.

Au-delà de cet accompagnement ponctuel, notre Etablissement a positionné ses Responsables pédagogiques comme des « super profs » capables de co-construire avec vous une scolarité sur-mesure.
En somme, le Responsable pédagogique est votre premier point de contact identifié, à même de vous guider et de répondre à vos différents questionnements.

Votre Responsable pédagogique est la personne en charge du suivi de la scolarité des élèves.
Il est tout naturellement votre premier référent : une question, un doute, une incompréhension ? Votre Responsable pédagogique est là pour vous écouter et vous orienter. Autant que nécessaire et sans aucun surcoût.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.

QUEL EST SON RÔLE ?

Orienter les parents et les élèves.
Proposer la mise en place d'un accompagnement individualisé de l'élève.
Faire évoluer les outils pédagogiques.
Encadrer et coordonner les différents professeurs.

VOS PROFESSEURS CORRECTEURS

Notre Etablissement a choisi de s'entourer de professeurs diplômés et expérimentés, parce qu'eux seuls ont une parfaite connaissance de ce qu'est un élève et parce qu'eux seuls maîtrisent les attendus de leur discipline. En lien direct avec votre Responsable pédagogique, ils prendront en compte les spécificités de l'élève dans leur correction. Volontairement bienveillants, leur correction sera néanmoins juste, pour mieux progresser.

QUAND PUIS-JE LE JOINDRE ?

Une question sur sa correction ?

- faites un mail ou téléphonez à votre correcteur et demandez-lui d'être recontacté en lui laissant un message avec votre nom, celui de votre enfant et votre numéro.
- autrement pour une réponse en temps réel, appelez votre Responsable pédagogique.

LE BUREAU DE LA SCOLARITÉ

Placé sous la direction d'Elena COZZANI, le Bureau de la Scolarité vous orientera et vous guidera dans vos démarches administratives. En connaissance parfaite du fonctionnement de l'Etablissement, ces référents administratifs sauront solutionner vos problématiques et, au besoin, vous rediriger vers le bon interlocuteur.

QUAND PUIS-JE LE JOINDRE ?

Du lundi au vendredi : horaires disponibles sur votre carnet de route et sur PoulPi.
04.67.34.03.00
scolarite@cours-pi.com

LE SOMMAIRE

OBJECTIFS

- Passer de la représentation d'une base dans une autre.
- Évaluer le nombre de bits nécessaires à l'écriture en base 2 d'un entier, de la somme ou du produit de deux nombres entiers.
- Utiliser le complément à 2.
- Calculer sur quelques exemples la représentation de nombres réels.
- Dresser la table d'une expression booléenne.
- Identifier l'intérêt des différents systèmes d'encodage.
- Convertir un fichier texte dans différents formats d'encodage.

COMPÉTENCES VISÉES

- Comment sont représentés les entiers naturels et relatifs pour un ordinateur.
- Comment sont représentés les nombres réels.
- Le rôle des booléens et des expressions booléennes.
- L'utilité des différents formats d'encodage de texte.

CHAPITRE 2. Représentation des données : types construits 23

OBJECTIFS

- L'utilisation des tuples, ou p-uplets.
- L'utilisation des tableaux, ou listes.
- L'utilisation des p-uplets nommés et des dictionnaires.

COMPÉTENCES VISÉES

- Écrire une fonction renvoyant un p-uplet de valeurs.
- Lire et modifier les éléments d'un tableau grâce à leurs index.
- Construire un tableau par compréhension.
- Utiliser des tableaux de tableaux pour représenter des matrices : notation a [i] [j].
- Itérer sur les éléments d'un tableau.
- Construire une entrée de dictionnaire.
- Itérer sur les éléments d'un dictionnaire.

CHAPITRE 3. Traitement de données en tables 47

OBJECTIFS

- Importer une table depuis un fichier texte tabulé ou un fichier CSV.
- Rechercher les lignes d'une table vérifiant des critères exprimés en logique propositionnelle.
- Trier une table suivant une colonne.
- Construire une nouvelle table en combinant les données de deux tables.

COMPÉTENCES VISÉES

- Apprendre à distinguer ce qui est exécuté sur le client et sur le serveur.
- Pouvoir reconnaître quand et pourquoi la transmission est chiffrée.
- Savoir analyser le fonctionnement d'un formulaire simple.
- Apprendre à distinguer les transmissions par les requêtes POST ou GET.
- Savoir identifier les composants permettant d'interagir avec une application web.
- Savoir identifier les évènements déclenchables par les scripts clients.

CORRIGÉS à vous de jouer et exercices 69

SUGGESTIONS CULTURELLES

SITES RESSOURCES

- **www.numérama.com**
- **www.meta-media.fr**
- **www.pixees.fr**
- **Lumni** *: vidéos, articles, quizz*
  www.lumni.fr/lycee/premiere/voie-generale/nsi-numerique-et-sciences-informatiques

PODCASTS

- **Le code a changé** *France Inter*
- **La vie numérique** *France Culture*
- **Culture numérique**

ESSAIS

- **La pensée informatique** *Gérard Berry*
- **La cryptologie au coeur du numérique** *Jacques Stern*
- **Designing an Internet** *David D. Clark*

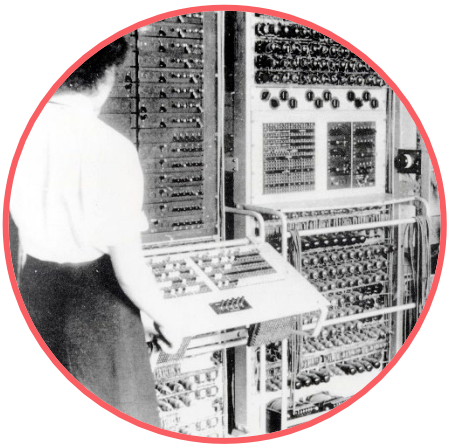
DOCUMENTAIRES AUDIOVISUELS

- **Une contre histoire de l'internet** *Sylvain Bergère*
- **Helvetica** *Gary Hustwit*
- **Citizenfour** *Laura Poitras*
- **Les ordinateurs du passé** *Le Vortex*


Le Cours

CHAPITRE 1

REPRÉSENTATION DES DONNÉES : TYPES ET VALEURS DE BASE



Les ordinateurs excellent dans les tâches que nous leur confions, à condition de leur expliquer clairement ce que nous attendons d'eux ! Pour cela, il nous est nécessaire de passer par des langages de programmation qui font le relai entre ce que nous sommes capables de comprendre et d'énoncer d'une part, et le langage machine d'autre part. On parle alors d'algorithmes, de programmes, d'applications… mais tout comme vous n'êtes capables de réfléchir à un sujet qu'à partir de connaissances initiales, un programme ne peut fonctionner qu'à partir d'informations numériques, qu'on appelle communément données. Elles peuvent prendre de nombreuses formes et devenir très complexes.

Nous allons étudier dans ce chapitre les formes les plus basiques de données informatiques. Celles à partir desquelles les autres pourront être ensuite formées !

OBJECTIFS

- Passer de la représentation d'une base dans une autre.
- Évaluer le nombre de bits nécessaires à l'écriture en base 2 d'un entier, de la somme ou du produit de deux nombres entiers.
- Utiliser le complément à 2.
- Calculer sur quelques exemples la représentation de nombres réels.
- Dresser la table d'une expression booléenne.
- Identifier l'intérêt des différents systèmes d'encodage.
- Convertir un fichier texte dans différents formats d'encodage.

COMPÉTENCES VISÉES

- Comment sont représentés les entiers naturels et relatifs pour un ordinateur.
- Comment sont représentés les nombres réels.
- Le rôle des booléens et des expressions booléennes.
- L'utilité des différents formats d'encodage de texte.

PRÉ-REQUIS

- Étude du premier module de NSI 1ère (notamment sur la question du binaire et des portes logiques).

MATERIEL NECESSAIRE

- Un ordinateur permettant d'exécuter des commandes et programmes Python.

Première approche
À la découverte du réseau

Tron est un film de science-fiction américain réalisé par Steven Lisberger, sorti en 1982 et produit par les Studios Walt Disney.

POUR ALLER PLUS LOIN

TRON – Un film de Steven Lisberger (1982)

Le film, qui traite du monde informatique et présente à la fois une plongée dans le monde virtuel et l'existence d'une intelligence artificielle, est célèbre pour traiter du monde informatique alors qu'à l'époque, la souris à boule faisait à peine ses débuts.

A voir en DVD ou sur toutes les plateformes légales de streaming

Le film plonge ses personnages dans un univers virtuel où chaque élément d'un ordinateur prend vie, d'une manière ou d'une autre ! Il s'agit du premier long métrage ayant eu recours de manière intensive à des scènes réalisés par ordinateur. Chaque programme y est représenté par une personne… L'un des héros du film est accompagné d'un « bit », une sorte de « créature » volante qui lui donne des informations et des conseils.

Bit est en général dans cet état, silencieux

Parfois, il s'écrie « oui » et change de forme pour adopter celle-ci

Il peut aussi dire « non » et prend alors cette forme :

Après avoir pris brièvement l'état oui ou non, il reprend vite son état initial.

La suite de Tron en jeu vidéo, réalisée par Buena Vista Interactiveen 2003, propose le même type de contexte mais dans un univers plus moderne. Le compagnon « Bit » a évolué ! Désormais, le héros est accompagné par « Byte » (« Octet » en français).

Celui-ci n'est plus limité aux mots « oui » et « non », mais peut parler avec des phases complexes (sur un ton monotone).

À VOUS DE JOUER 1

Répondez aux questions suivantes en prenant appui sur les documents précédents.

1. Que pensez-vous du personnage Bit ? Vous semble-t-il cohérent par rapport à l'univers dans lequel il évolue ?

2. Même question pour Byte.

01 TYPES ET VALEURS DE BASE

Représentation des entiers naturels

Nous allons partir ici de quelques notions mathématiques. Les nombres peuvent appartenir à différents ensembles. Nous avons celui des entiers, dans lequel on compte le plus souvent : 1, 2, 3, 4, etc. Nous utilisons aussi fréquemment l'ensemble des réels : lorsque vous achetez une baguette qui coûte 80 centimes (80 est un entier) vous pouvez écrire sur votre carnet de dépenses que vous avez payé 0,80 euros (et ça c'est un réel, avec des chiffres après la virgule).

Au quotidien, nous avons tellement l'habitude de passe d'un type d'ensemble à un autre que nous n'y prêtons pas attention. En informatique, tout dépend du langage employé. Certains langages « devinent » le type de donnée que l'on veut mettre dans une variable (une variable est un emplacement mémoire dans lequel on stocke une information). On parle alors de langages non typés ou faiblement typés. Cela a un aspect pratique mais peut générer parfois des approximations ou des erreurs d'interprétation. D'autres langages obligent l'utilisateur à préciser le type de donnée contenu dans la variable. Il s'agit alors de langages fortement typés. Ils peuvent paraître plus contraignants, mais ils permettent de repérer plus facilement certaines erreurs lors de l'exécution d'un programme.

Dans les deux cas, celui du langage faiblement typé (comme Python) ou du langage fortement typé (comme Java), c'est la façon dont on code qui change, mais pour la machine chaque donnée (stockée dans une variable) correspond au final à un type bien défini.

Pour ce qui est des entiers naturels, nous en utilisons chaque jour sur une base décimale (avec dix signes ou chiffres pour les représenter). Mais d'autres bases sont possibles, et certaines sont utiles en informatique.

EXERCICE 01

Représentation des entiers naturels dans différentes bases

Remplissez le tableau suivant.

Essayez de remplir ce tableau à l'aide de calculs manuels si vous le pouvez. Sinon, utilisez un outil en ligne ou une calculatrice pour cela. D'autres exercices un peu plus loin vous permettront de procéder par étapes à des conversions sur le binaire ou l'hexadécimal.

Représentation / Base	Signes utilisés	Entier, exemple 1	Entier, exemple 2
Binaire	**0,1**	**110111**	**100111110**
Ternaire		**2001**	
Octale		**67**	
Décimale	**0,1,2,3,4,5,6,7,8,9**	**55**	
Hexadécimale	**0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**	**37**	

COMPTER EN BINAIRE : RAPPEL

Si le rang des unités ne peut accepter que deux symboles, 0 ou 1, il est impossible d'écrire 2 et donc on doit recourir au rang suivant. On compte ainsi en binaire de la sorte : 0, 1, 10, 11, 100, etc.

Voilà ce que ce comptage donne si on le compare au comptage décimal, jusqu'à 9 :

Valeur décimale	Valeur binaire	Commentaire
0	0	
1	1	On est toujours sur un chiffre présent dans les deux systèmes.
2	10	Ici, les unités ont atteint leur maximum, on passe donc au rang suivant.
3	11	Le premier rang peut passer de 0 à 1.
4	100	Deux rangs pleins donc on les vide pour remplir le troisième.
5	101	On continue sur le même principe.
6	110	
7	111	
8	1000	
9	1001	

Et ainsi de suite ! Chaque rang (en couleur dans le tableau) correspond à une puissance de deux.

Puissance	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Décimal	128	64	32	16	8	4	2	1
Binaire	10000000	1000000	100000	10000	1000	100	10	1

Ainsi, le nombre binaire 1010 correspond en binaire à la somme de 2^3 et 2^1, soit 8 + 2, soit 10.

EXERCICE 02

Conversions entre décimal et binaire.

1. Convertissez du binaire vers le décimal les nombres suivants : $110\,101_2$ / $001\,111_2$

2. Convertissez du décimal vers le binaire les nombres suivants : 36_{10} / 125_{10}

COMPTER EN HEXADÉCIMAL

Quelle que soit la base, le principe est toujours le même, on change de rang lorsque le rang actuel arrive à son dernier signe. On change très vite de rang avec le binaire, qui ne dispose que de deux signes. Mais à l'inverse, on reste plus longtemps sur chaque rang avec l'hexadécimal qui contient seize signes !
On compte donc de la sorte : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, et… A ! Puis B, C, D, E et F. Après ça ? On est à court de signe, donc on écrit 10, puis 11, etc. 10 en hexadécimal équivaut à 16 en décimal.

Puissance	16^4	16^3	16^2	16^1	16^0
Décimal	65536	4096	256	16	1
Hexadécimal	10000	1000	100	10	1

<ASTUCE PRATIQUE>

Notation sur différentes bases.

Comment savoir si le nombre 100 est en décimal (prononcé « cent ») ou en binaire (prononcé « un zéro un ») ? Vous pouvez noter la base d'un nombre en indice à côté de lui : 1010010_2. Le même nombre en base 16 s'écrit 52_{16} . A noter que vous trouverez aussi parfois la notation avec parenthèses : (1010010).

EXERCICE 03

Conversions entre décimal et hexadécimal.

1. Convertissez de l'hexadécimal vers le décimal les nombres suivants : $1\,1011_{16}$/ $5E_{16}$

2. Convertissez du décimal vers le binaire les nombres suivants : 36_{10} / 301_{10}

EXERCICE 04

Récapitulatif du comptage dans différentes bases.

Voici un tableau permettant de voir côte à côte les différents modes de comptages les plus courants en informatique. Il va jusqu'à 2410 mais vous pourriez le continuer encore au-delà !

Remplissez les cases vides, sachant qu'on se contente d'ajouter une unité à chaque ligne.

Binaire (base 2)	Décimal (base 10)	Octal (base 8)	Hexadécimal (base 16)
0	**0**	**0**	**0**
1	**1**	**1**	**1**
10	**2**	**2**	**2**

UTILISATION DE PYTHON

Si ce n'est pas déjà fait, installez Python. Soit le dernier Python de base, soit Edupython. Un document complémentaire est disponible pour vous accompagner dans cette installation.

La fonction bin() permet de convertir un entier naturel en binaire. Le résultat est préfixé par « 0b ».
De la même façon, la fonction hex() renvoie une chaîne de caractère qui n'est autre que la représentation de cet entier en hexadécimal. Ce résultat est prefixé par « 0x ».

Ainsi, bin(18) renvoie 0b10010 et hex(18) renvoie 0x12.

La fonction print() permet d'afficher des données dans la console.
Exemple : print(bin(18))

Conversions dans Python : depuis le décimal.

1. Quel est le résultat du programme suivant ? :

```
print(bin(3422))
print(hex(98329))
```

2. Quel est le résultat du programme suivant ? :

```
print(oct(3422))
print(oct(98329))
```

3. D'après vous, quel est l'utilité de la fonction oct() ?

De la même façon, on peut effectuer les opérations inverses avec la fonction int(). Celle-ci utilise deux arguments : une chaîne de caractères représentant un entier dans une base de données, puis un entier représentant cette base.
Exemple : int('0b10010', 2) renvoie 18.

Conversions dans Python : vers le décimal.

Quel est le résultat du programme suivant ? :

```
print(int('0b1110101', 2))
print(int('0o743', 8))
print(int('0x4d5', 16))
```

TYPES ET VALEURS DE BASE
Représentation des entiers relatifs

LE NÉGATIF EN BINAIRE : COMPLÉMENT À DEUX

Le comptage que nous avons vu ne permet pas pour un ordinateur de gérer les entiers relatifs. En effet, on ne peut pas se contenter de mettre le signe « moins » devant un nombre binaire informatique si l'ordinateur n'interprète que des signaux électriques de type « on » ou « off » via des 0 et des 1.
Pour cela, on a recours à ce qu'on appelle le complément à deux. Cette méthode ne fonctionne que sur des nombres binaires ayant le même nombre de bits (généralement un multiple de 4). Parmi les bits utilisés, on désigne alors le bit de poids fort (le plus à gauche) comme étant le bit de signe, à savoir celui qui sert à représenter le signe du nombre représenté (positif ou négatif).
Très bien ! Il suffirait donc, sur 8 bits, d'écrire ceci :

00000011 = +3 en décimal
et
10000011 = -3 en décimal

Eh bien ce ne sera pas toujours pratique ! Car si l'on procède ainsi, l'addition binaire ne fonctionne plus. Voyez : en additionnant 3 et -3 on devrait arriver à zéro, mais l'addition binaire 00000011 + 10000011 ne sera pas d'accord !

C'est pour cela qu'on utilise dans certains cas le complément à deux :

- les nombres positifs sont représentés de façon standard ;
- les nombres négatifs passent par une inversion de valeur pour chaque bit, puis l'ajout de 1 au total.

Exemple, pour coder -8 procédez de la manière suivante :

Prenez le nombre positif 8 :	0000 1000
Inversez les bits :	1111 0111
Ajoutez 1 :	1111 1000

(on commence par l'unité qui ne peut pas accueillir 2, la retenue se déplace donc de rang en rang jusqu'au rang à 0 qui prend la valeur 1)

De même pour obtenir la valeur -43 :

Démarrage sur 43 :	0010 1011
Inversion :	1101 0100
Ajout de 1 :	1101 0101

Notons que le bit de signe (le premier bit à gauche) est automatique mis à 1 par l'opération d'inversion.

Testons une addition pour voir si le problème évoqué plus haut est résolu.

Décimal : 2 + (-8) = -6
Binaire en complément à deux sur 8 bits : 0000 0010 + 1111 1000 = 1111 1010

Le bit de signe montre qu'il s'agit d'un nombre négatif, donc on effectue un complément à deux pour retrouver sa valeur absolue.

Inversion des bits :	0000 0101
Ajout de 1 :	0000 0110

On tombe bien sur une valeur de 6, le bit de signe était à 1 donc on sait nous avions le nombre -6.

EXERCICE 07

Complément à deux.

1. Complétez le tableau suivant, dans lequel il s'agit de trouver les valeurs binaires en complément à deux codées sur 8 bits pour les décimaux donnés dans la première colonne. Les premières lignes préremplies sont des rappels des exemples vus plus haut.
   Note : rappelez-vous que pour une lecture et une écriture plus facile, on peut séparer les 8 bits en deux blocs de 4 bits séparés par un espace.

Décimal à convertir	Représentation en complément à deux sur 8 bits
2	**0000 0010**
-8	**1111 1000**
-43	**1101 0101**
-2	
0	
1	
64	
-64	
127	
-127	
128	

2. Un complément à deux sur 4 bits fonctionnerait sur le même principe. Dans cet exercice, vous aviez en question 1 un bit de signe et sept bits pour représenter la valeur, donc une plage de valeurs allant de -128 à 127. Sur 4 bits vous aurez un bit de signe et trois bits de valeur, pour une plage allant de -8 à 7).
   Remplissez ce tableau pour le vérifier.

Décimal à convertir	Représentation en complément à deux sur 4 bits
7	
2	
-5	
-7	
-8	

ESPACE MÉMOIRE ALLOUÉ AUX ENTIERS

Avant de terminer sur les entiers, qu'ils soient positifs ou négatifs, examinons d'un peu plus près la place qu'ils prennent en mémoire. Cela dépend en fait des machines et surtout des langages de programmation.

Commençons par observer comment le langage Java stocke ce type de données.
Java propose quatre types d'entiers :

Type	Nombre d'octets	Plage de valeurs possibles
byte	1	-128 à 127
short	2	-32 768 à 32 767
int	4	-2 147 483 648 à 2 147 483 647
long	8	-9 223 372 036 854 775 808 à 9 223 372 036 854 775 807

On voit ici une logique se dégager. Pour les raisons que nous avons vues précédemment (codage d'un nombre en binaire et complément à 2), la plage proposée pour un type **byte** utilisant un seul octet en mémoire est cohérente.

Ajoutez un octet pour avoir un type **short** et vous gagnez 8 bits de codage binaire, vous donnant une plage plus large. C

Mais des entiers d'environ 32 000 en valeur absolue restent facilement trop restreints, le type **int** codé sur 4 octets est donc lui aussi couramment employé.

Dans les cas plus extrêmes, un entier de type **long** pourra être utilisé. Les 8 bits qu'il utilise permettent de coder des valeurs très grandes, grâce à la puissance de l'exponentielle.

Et dans Python ? Son utilisation des ressources est plus flexible mais aussi plus gourmande. Essayez d'exécuter le code suivant, il vous indiquera combien d'octets sont utilisé pour différents entiers.

```
import sys

# nombre d'octets utilisés
# pour un type int de base
print(sys.getsizeof(int()))

# nb d'octets pour l'entier 0
print(sys.getsizeof(0))

# nb d'octets pour l'entier 1
print(sys.getsizeof(1))

# nb d'octets pour l'entier 1234567890
print(sys.getsizeof(1234567890))

# nb d'octets pour l'entier 1234567890 au carré
print(sys.getsizeof(1234567890**2))
```

TYPES ET VALEURS DE BASE

Nombres flottants

Nous avons vu le cas des entiers. Qu'en est-il des réels ? Ils peuvent comporter des chiffres après la virgule, et on ne peut donc pas les stocker de la même façon.
En informatique, on les appelle nombres flottants.

Autre point de comparaison avec les entiers :

- entre deux bornes quelconques pour les entiers, il y a un nombre fini d'autres entiers (par exemple, entre 2 et 5 exclus, on trouve deux nombres, à savoir 3 et 4) ;
- entre deux bornes quelconques pour les réels, il y en a une infinité, puisqu'on peut toujours aller plus loin après la virgule.

Pourtant, même s'il y a une infinité de réels ne serait-ce qu'entre 1 et 2, il n'est pas possible de stocker physiquement dans un ordinateur un nombre infini de chiffres après la virgule.

Il en résulte que pour certains nombres, le stockage est exact et précis (par exemple 5 et 5,5), tandis que pour d'autres on ne peut stocker qu'une approximation (c'est le cas notamment pour 1/3 qui contient une infinité de 3 après la virgule, ou pour le nombre Pi qui contient toutes sortes de chiffres après la virgule, toujours à l'infini).

Voyons comment un ordinateur peut comprendre ce genre de nombres. Nous allons procéder par étapes.

ESPACE MÉMOIRE ALLOUÉ AUX ENTIERS

Nous avons vu comment chaque rang en binaire correspond à une puissance de deux. Il est possible d'étendre cette correspondance de l'autre côté de la virgule. Cela donne ceci :

Puissance	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Décimal	8	4	2	1	1/2	1/4	1/8	1/16
Binaire	1000	100	10	1	,1	,01	,001	,0001

Le nombre 100,101 en binaire peut ainsi être décomposé et converti en décimal de la sorte :

$2^2 + 2^{-1} + 2^{-3}$
$= 4 + 1/2 + 1/8$
$= 4 + 0,5 + 0,125$
$= 4,625$

Bien sûr, on peut aller plus loin que 2^{-4}. Par exemple voici les valeurs jusqu'à 2^{-8} :

2^{-1}	1/2	0,5
2^{-2}	1/4	0,25
2^{-3}	1/8	0,125
2^{-4}	1/16	0,0625
2^{-5}	1/32	0,03125
2^{-6}	1/64	0,015625
2^{-7}	1/128	0,0078125
2^{-8}	1/256	0,00390625

EXERCICE 08

Conversions entre décimal et hexadécimal.

1. Convertissez en décimal les nombres suivants : $1000,0101_2$ / $1101,1101_2$

2. Convertissez en binaire les nombres suivants : $33,875_{10}$ / $260,1875_{10}$

NOTATION SCIENTIFIQUE : RAPPEL

Les exemples que nous venons de voir utilisaient des virgules, mais en langage machine (constitué uniquement de 0 et de 1) ce signe n'existe pas. Comment dans ce cas représenter la limite entre les chiffres avant et après la virgule ?

Nous allons utiliser une méthode qui n'est pas sans rappeler la notation scientifique que vous avez probablement croisé en cours de mathématiques et qui est très utilisé dans les calculatrices. Pour rappel, la notation scientifique fonctionne de la manière suivante : 5 430 000 = **5,43 * 10^6**

La première partie de cette notation (dans notre exemple 5,43) est appelée **mantisse**.
La seconde partie (ici 10^6) est appelé **exposant**.

La mantisse est toujours ajustée de sorte qu'elle ne contienne qu'un seul chiffre (en dehors de zéro) à gauche de la virgule. L'exposant nous permet de connaître l'étendue du décalage à effectuer pour retrouver le nombre en notation classique.
On peut d'ailleurs noter que si l'exposant est positif on décale la virgule vers la droite. Mais si l'exposant est négatif, on décale la virgule vers la gauche.

Ainsi, sur ce modèle : 0,00438 = **4,38 * 10^{-3}**

Notation scientifique.

1. Ecrivez en notation scientifique les nombres suivants : :

3
4 232
89 303 910
0,021
0,00504

2. Ecrivez en notation décimale les nombres suivants : :

5,322 * 10^7
3,890832 *10^{11}
2,14232 * 10^2
9,32 * 10^0
6,4 * 10^1

VIRGULE FLOTTANTE : LA NORME IEEE 754

En gardant à l'esprit ce que nous venons de revoir sur la notation scientifique, explorons un type de données informatique défini par la norme IEEE 754 (datant de 1985). Cette norme définie deux formats :

- un format codé sur 32 bits (appelé « simple précision » ou binary32) ;
- un format codé sur 64 bits (appelé « double précision » ou binary64).

Nous nous attarderons surtout sur le premier, le second étant simplement un format étendu qui, grâce à son utilisation de bits supplémentaires, permet de stocker les nombres de manière plus étendue.

NOMBRE À VIRGULE FLOTTANTE À PRÉCISION SIMPLE

Dans les langages de programmation, ils sont souvent désignés par le type **float** ou **real**.
Ils sont longs de 4 octets (ou 4 bytes en anglais), donc 4 fois 8 bits, c'est-à-dire 32 bits, comme nous l'avons vu précédemment.

<ASTUCE PRATIQUE>
Bytes et bits !
Attention à ne pas confondre ces deux mots qui se ressemblent mais n'ont pas la même signification. Byte est l'équivalent du mot français « octet » ce qui désigne un groupe de 8 bits. Bit est un mot identique dans les deux langues.

Exemple de décodage

Voici un nombre en virgule flottante binary32 décomposé en 3 parties :

Signe	Exposant	Mantisse
1	**1000 1001**	**101 1010 0000 0000 0000 0000**

Traduisons-le en notation décimale.

Le signe indique ici un nombre négatif.

L'exposant réel est obtenu en soustrayant 127 à la valeur du champ exposant (ce procédé permet de gérer les exposants négatifs en plus des positifs).

$1000\ 1001_2 = 137_{10}$
$137 - 127 = 10$
Exposant réel : 2^{10}
(n'oublions pas que nous sommes en base 2, donc un exposant de 2^{10} et non 10^{10})

La mantisse que nous cherchons est codé sur 23 bits mais correspond à une valeur de 24 bits donc le premier bit serait fixe et codé à 1 (qu'on appelle aussi bit caché). C'est un bit implicite dont la présence est expliquée par le fait que nous travaillons ici en notation scientifique. La mantisse ne commence donc jamais par zéro.

La mantisse, avec le premier bit implicite, est donc :
1,101 1100 0000 0000 0000 0000

Note : la virgule n'est pas stockée ainsi dans la mémoire, mais va nous permettre de la recalculer en décimal manuellement pour vérification.

Cette mantisse peut être décomposée de la sorte :
$2^0 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6}$
$= 1 + 0{,}5 + 0{,}125 + 0{,}0625 + 0{,}015625$
$= 1{,}703125$

Avec l'exposant trouvé plus haut, cela donne le nombre suivant :
$1{,}703125 * 2^{10}$
$= 1{,}703125 * 1024$
$= 1744$

Et n'oublions pas que nous étions sur un nombre négatif. Donc le nombre stocké est : -1744

EXERCICE 10

Notation scientifique.

1. Soit le nombre stocké en binary32 de la manière suivante : 0 0111 1011 011 1110 0000 0000 0000 0000
   Calculez sa valeur décimale. Vous pouvez vous aider d'outils en ligne ou d'une calculatrice scientifique.

2. Soit le nombre stocké en binary32 de la manière suivante : 1 0111 0111 101 0100 0000 0000 0000 0000
   Calculez sa valeur décimale. Vous pouvez vous aider d'outils en ligne ou d'une calculatrice scientifique.

3. Soit le nombre stocké en binary32 de la manière suivante : $42CC0000_{16}$
   Calculez sa valeur décimale. Vous pouvez vous aider d'outils en ligne ou d'une calculatrice scientifique.
   Il est stocké ici en hexadécimal (plus court et plus pratique), et donc à convertir d'abord en binaire.

EXERCICE 11

Nombres à virgule dans Python.

1. Exécutez dans Python : print (1/3). Interprétez le résultat.

2. Exécutez dans Python : print ((0.2 + 0.1) – 0.3). Interprétez le résultat.

AVERTISSEMENT CONCERNANT LES FLOTTANTS

Nous avons déjà évoqué le fait qu'en raison des limitations mémoire, les nombres flottants que nous manipulons sont parfois tronqués et soumis à des approximations.

Tapez tout simplement ceci dans une console Python : >>> 0.1 + 0.2

Ou exécutez le programme équivalent : print (0.1 + 0.2)

Quel est le résultat ?
Indice : ce n'est pas 0.3 malheureusement.

Pour cette raison, il faut rester vigilant dans l'emploi des **floats**, et par exemple éviter de tester l'égalité entre deux flottants.

04 TYPES ET VALEURS DE BASE

Booléens

Le module de cours NSI 1re numéro 1 consacré aux architectures matérielles et logicielles explore en détail l'utilisation des booléens et des portes logiques (AND, OR, etc.).

Rappelons ici simplement que ce type de données n'a pour objectif que de stocker deux valeurs possibles : **VRAI** ou **FAUX**. Dans les langages de programmation (presque toujours en anglais), on utilise les valeurs **True** et **False** (l'importance de la casse dépend du langage employé).

Ces valeurs ne sont pas des chaînes de caractères. Il s'agit plutôt, plus simplement, des deux valeurs possibles d'un bit.

VRAI = TRUE = 1
FAUX = FALSE = 0

Les **portes logiques** font référence à des agencements de transistors (ou leurs équivalents modernes, miniaturisés) qui permettent de réaliser des fonctions acceptant des paramètres d'entrée, pour obtenir une valeur de sortie.

Par exemple, la porte OR (en français « ou ») renvoie 1 si au moins un de ses deux paramètres d'entrée est égal à 1.

La porte AND (en français « et ») renvoie 1 si et seulement si les deux paramètres d'entrées sont égaux à 1.

La porte NOT (en français « non ») inverse la valeur de son seul paramètre d'entrée.

La porte XOR (en français « ou exclusif ») renvoie 1 si un seul de ses paramètres d'entrée est égal à 1.

Il existe d'autres portes : NAND, NOR et XNOR.

Booléens et portes logiques de base.
Si A est vrai, et B est faux :

1. Que vaut A OR B ?

2. Que vaut A AND B ?

3. Que vaut NOT A ?

4. Que vaut NOT B ?

TABLE DE VERITÉ D'UNE EXPRESSION BOOLÉENNE

On peut noter le comportement d'une porte logique à l'aide d'une table de vérité. Examinons le cas pour l'opérateur AND.

Soit l'expression suivante :

RES = var1 AND var2

On peut y associer la table de vérité qui suit :

Paramètres		Résultat
var1	**var2**	**RES**
0	0	0
0	1	0
1	0	0
1	1	1

RES ne sera « vrai » (égal à 1) que si var1 et var2 sont aussi égaux à 1.

EXERCICE 13

Table de vérité pour trois paramètres d'entrée.

Soit l'expression suivante : RES = (var1 OR var2) AND var3

Établissez sa table de vérité.

Paramètres		Résultat
var1	var2	RES

05

TYPES ET VALEURS DE BASE

Texte : encodages et tables de caractères

Nous avons vu comment sont stockés différentes sortes de nombres, ainsi que les booléens (très pratique en algorithmie), mais qu'en est-il du texte ? Les langages de programmation les gèrent sous des types qu'ils nomment en général **STRING** (avec des variantes, mais contentons-nous de cela pour l'instant). En français « chaîne » de caractère. Mais comment sont stockés les caractères eux-mêmes, avec leurs spécificités (accents par exemple), dans un fichier texte ?

L'ENCODAGE ASCII

Utilisé durant de nombreuses années depuis son introduction en 1963, le code ASCII (American Standard Code for Information Interchange) permettait de stocker l'alphabet occidental (non accentué) ainsi que plusieurs caractères utilitaires (tabulation, retour à la ligne, etc.), sur un seul octet.
Dans cet octet, 7 bits étaient dédiés à l'encodage du caractère, avec donc 128 valeurs possibles.
Ne disposant pas de caractères accentués, il était tout à fait suffisant pour écrire en anglais mais lacunaire pour le français (on notera d'ailleurs que les adresses e-mail, toujours codées en ASCII, n'acceptent pas les accents). Si on ne compte pas les caractères « invisibles », le code ASCII propose 95 caractères imprimables :

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```

Le code ASCII a, au cours des années, connu des améliorations et extensions.

EXERCICE 14

Table ASCII.

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

1. Indiquez quel est le code décimal correspondant à :
   - la lettre A majuscule ;
   - la lettre A minuscule ;
   - l'espacement.

2. Indiquez quel est le caractère encodé par :
   - la valeur décimale 37 ;
   - la valeur hexadécimale 59 ;
   - la valeur binaire 1010001.

ISO-8859-1 (OU LATIN1)

Fonctionnant sur le même principe que l'ASCII, cette norme utilisait 8 bits effectifs au lieu de 7, portant le nombre de caractères possibles à 256. C'est cette norme que nous avons utilisé en France après l'ASCII, car elle contenait les accents dont nous avions besoin.

Malheureusement, chaque langue avait ses propres besoins. Plus de 50 normes de ce genre ont donc été créés par l'organisme ISO en charge de ce travail. Cela rendait parfois la communication entre machines difficile.

UNICODE

Depuis 1991, cette nouvelle norme comprend dans un même ensemble tous les caractères possibles. Elle peut être codée de différentes façons, mais le système UTF-8 est le plus courant (notamment sur le web).

Grâce à UTF-8, on peut encoder un caractère suivant un nombre variable d'octets. Les caractères les plus courants n'utilisent qu'un octet. Les caractères plus inhabituels peuvent nécessiter jusqu'à quatre octets. Cet encodage est en outre compatible avec le code ASCII.

<ASTUCE PRATIQUE>
Convertir des fichiers d'un encodage à un autre.
Avec un éditeur de texte comme Notepad++ (téléchargeable et utilisable gratuitement), vous pouvez expérimenter ces différents encodages.
Créez un fichier texte vide et tapez quelques mots dedans, contenant des caractères avec accents.
Puis utilisez le menu Encodage pour passer à d'autres encodages, ajouter d'autres mots avec accents, etc.
Naviguez entre les encodages suivants : ANSI, UTF-8, ISO-8859-1 (accessible via Encodage → Codage de caractères → Langues d'Europe occidentale).
Question : Constatez-vous des modifications dans votre texte au gré des changements d'encodage ?

Peut-être certains de vos caractères prennent une forme tout à fait cryptique, comme Ã©tÃ© !! C'est une conséquence directe du fait que tout encodage n'est pas fait pour accueillir n'importe quel caractère. La généralisation de l'UTF-8 a en grande partie réglé les problèmes liés à cet aspect. Lors des débuts du web, c'était une autre histoire…

Nous venons de passer en revue les types qui posent les fondements de la plupart des langages informatiques. Le binaire est de moins utilisé « tel quel » mais toujours très présent sous forme de booléens. Quant aux entiers et aux flottants, ils représentent une large part des données que vous aurez à manipuler lors de l'apprentissage de la programmation, et dans son utilisation professionnelle.

Ces types sont aujourd'hui relativement stables dans leur utilisation. C'est moins le cas pour les formats d'encodage du texte, qui évoluent sans cesse en raison de la complexité des langues humaines. Heureusement, ces changements vont généralement dans le bon sens : celui de la simplification et de l'universalité !

Vous pouvez maintenant faire et envoyer le devoir n°1