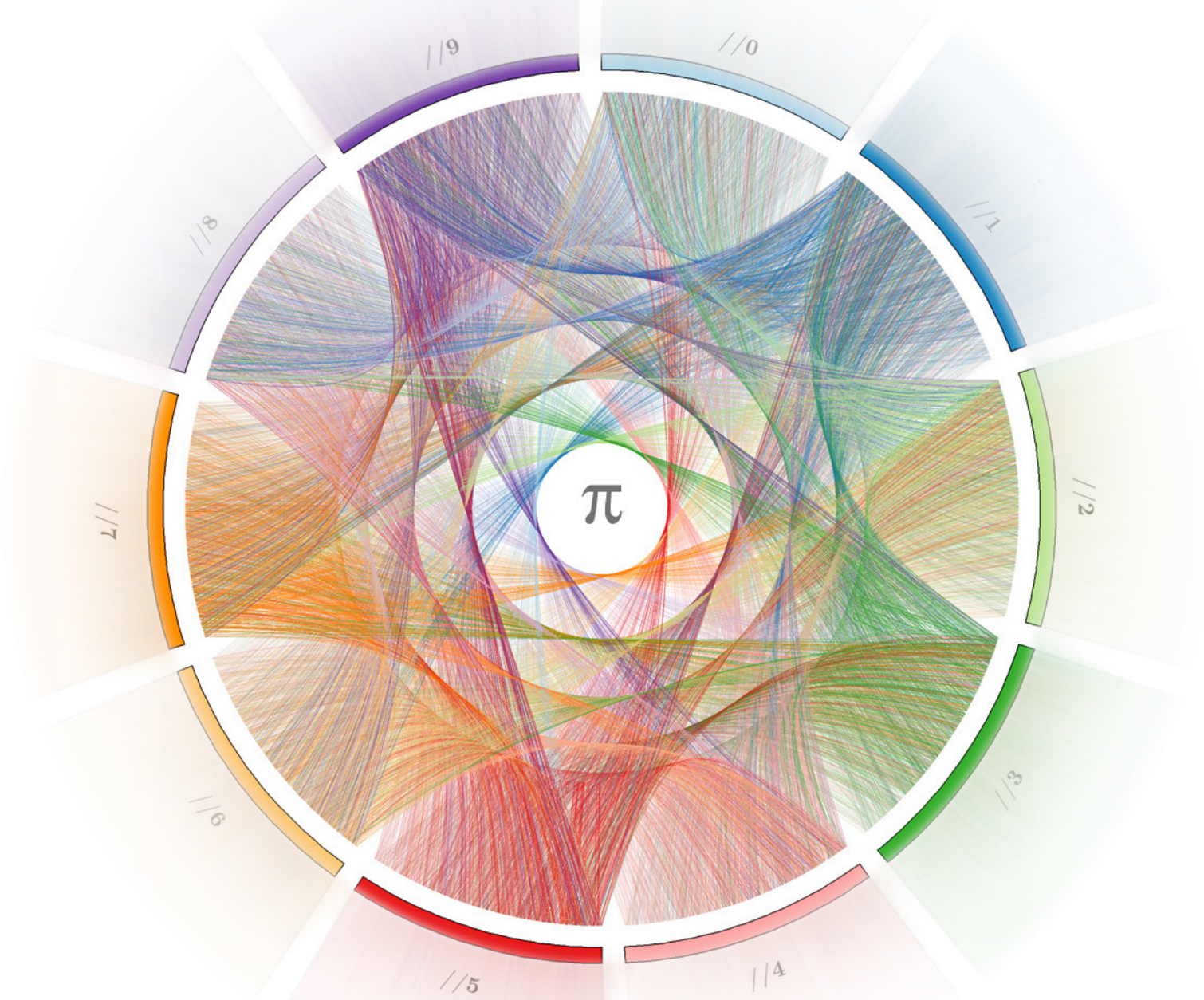
# Représentation des réels et caractères




GIF-1001 Ordinateurs : Structure et Applications, H2019

Jean-François Lalonde

Source image

# Rappel: décimal

Décortiquons 6 431,986…

| Position | 3 | 2 | 1 | 0 | , | -1 | -2 | -3 |
|---|---|---|---|---|---|---|---|---|
| Valeur | $10^3$ | $10^2$ | $10^1$ | $10^0$ | , | $10^{-1}$ | $10^{-2}$ | $10^{-3}$ |
| Symbole | 6 | 4 | 3 | 1 | , | 9 | 8 | 6 |
| = | 6000 | 400 | 30 | 1 | | 0,9 | 0,08 | 6 |

# Nombres rationnels

- Une possibilité (sur 16 bits):
  - mettons la virgule au milieu:

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Valeur | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ | , | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ | $2^{-5}$ | $2^{-6}$ | $2^{-7}$ | $2^{-8}$ |
| Position | $b_{15}$ | $b_{14}$ | $b_{13}$ | $b_{12}$ | $b_{11}$ | $b_{10}$ | $b_9$ | $b_8$ | , | $b_7$ | $b_6$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ |

    - 8 premiers bits: $2^0$ à $2^7$ (0 à 255)
    - 8 derniers bits: $2^{-1}$ à $2^{-8}$ (1/256 à 255/256, 0.00390625 à 0.99609375)

- Problèmes?
  - très limité!
    - quelle est la valeur maximale?
      - 255
    - quelle est la précision (ou la plus petite différence entre deux nombres)?
      - 1/256

# Rappel: décimal, notation scientifique

$$6\,500 = (+)\ 6{,}5 \times 10^{3}$$

# Rappel: décimal, notation scientifique

$$6\,500 = (\mathbf{+})\ 6{,}5 \times 10^{3}$$

**signe** (+): indique si le nombre est positif ou négatif

# Rappel: décimal, notation scientifique

$$6\,500 = (+)\ 6{,}5 \times \mathbf{10}^{3}$$

signe (+): indique si le nombre est positif ou négatif

**base** (10): décimale

# Rappel: décimal, notation scientifique

$$6\,500 = (+)\ 6{,}5 \times 10^{3}$$

signe (+): indique si le nombre est positif ou négatif

base (10): décimale

**exposant** (3): indique l'ordre de grandeur

# Rappel: décimal, notation scientifique

$$6\,500 = (+)\ \mathbf{6{,}5} \times 10^{3}$$

signe (+): indique si le nombre est positif ou négatif

base (10): décimale

exposant (3): indique l'ordre de grandeur

**significande** (6,5): détermine la précision de la valeur représentée

divisée en deux: **caractéristique** (6) et **mantisse** (0,5)

# Rappel: décimal, notation scientifique

$$6\,500 = (+)\ 6{,}5 \times 10^{3}$$

**signe** (+): indique si le nombre est positif ou négatif

**base** (10): décimale

**exposant** (3): indique l'ordre de grandeur

**significande** (6,5): détermine la précision de la valeur représentée

divisée en deux: **caractéristique** (6) et **mantisse** (0,5)

$$(\text{signe})\ \text{caractéristique,mantisse} \times \text{base}^{\text{exposant}}$$

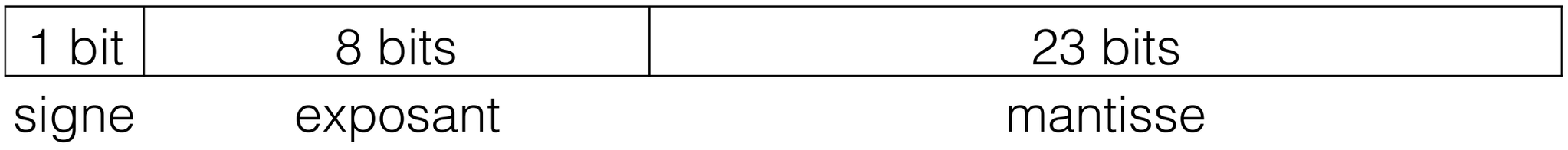
# Nombres rationnels, en binaire

- La norme IEEE 754 a été adoptée en 2008 pour les nombres rationnels sur 32, 64 et 128 bits

- Très similaire à la notation scientifique:

$$(\text{signe})\ 1,\text{mantisse} \times 2^{(\text{exposant}-127)}$$

- Sur 32 bits:
  - signe: un bit
  - base: 2, donc binaire. Comme cette base est toujours 2, on n'a pas besoin de la stocker (c'est implicite)
  - exposant (décalé): 8 bits (donc de 0 à 255), mais on soustrait 127, donc de -127 à +128



# IEEE754 vers décimal

Convertir 0x40D00000 (écrit en IEEE754) en décimal

- Tout d'abord, convertissons en binaire

| 4 | | 0 | | D | | 0 | | 0 | | 0 | | 0 | | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Séparons la chaîne de bits en sections correspondants aux champs de l'IEEE754 (signe, exposant, mantisse)

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- Analysons chaque champ indépendamment
  - Bit de signe: 0 donc positif (+)
  - Exposant: 0b10000001 = 129. 129 - 127 = 2.
  - Mantisse: 0b1010 000… $2^{-1} + 2^{-3} = 0,5 + 0,125 = 0,625$
- Résultat:
  - (+) 1,625 x $2^2 = 6,5$

(signe) 1,mantisse x $2^{(exposant-127)}$

| 1 bit | 8 bits | 23 bits |
|---|---|---|
| signe | exposant | mantisse |

# Exercice #1: IEEE754 vers décimal

Convertir 0x411A0000 (écrit en IEEE754) en décimal.

(signe) 1,mantisse x $2^{(exposant-127)}$

| 1 bit | 8 bits | 23 bits |
|---|---|---|
| signe | exposant | mantisse |

# Ex. 1, étape 1: hexadécimal vers binaire

Convertir 0x411A0000 (écrit en IEEE754) en décimal.

| 4 | 1 | 1 | A | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Hexadécimal | Binaire |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

# Ex. 1, étape 2: binaire vers décimal

Convertir 0x411A0000 (écrit en IEEE754) en décimal.

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- Bit de signe = 0, donc nombre positif

(+) 1,mantisse x $2^{(exposant-127)}$

- Exposant = 0b10000010 = 130. 130-127 = 3

(+) 1,mantisse x $2^{3}$

- Mantisse = 0b0011010…
  - = $1x2^{-3} + 1x2^{-4} + 1x2^{-6} = 0,125 + 0,0625 + 0,015625 = 0,203125$

(+) 1,203125 x $2^{3}$ = 9,625

(signe) 1,mantisse x $2^{(exposant-127)}$

| 1 bit | 8 bits | 23 bits |
|---|---|---|
| signe | exposant | mantisse |

# Décimal vers IEEE754

Convertir -16,125 en hexadécimal (IEEE754)

- Exprimons le nombre avec des puissances de 2

$16 = 2^4$ $0,125 = 2^{-3}$

- Convertissons en binaire

$-10000,001_b$

- Décalons la virgule vers la gauche

$-10000,001_b = -1,0000001_b \times 2^4$

- Identifions chacun des champs du format IEEE754
  - Signe - : bit à 1

| 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

  - Exposant: exposant - 127 = 4. 4+127 = 131 = $10000011_b$

| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

  - Mantisse: $00000010\ldots_b$

| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

# Décimal vers IEEE754 (suite)

Convertir -16,125 en hexadécimal (IEEE754)

| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- Regroupons les bits par groupes de 4

| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- Convertissons en hexadécimal

| C | 1 | 8 | 1 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

- Résultat: 0xC1810000

# Exercice #2: décimal vers IEEE754

Convertir 12,5 en binaire sur 32 bits avec IEEE 754.

Écrire la réponse en hexadécimal.

(signe) 1,mantisse x $2^{(exposant-127)}$

| 1 bit | 8 bits | 23 bits |
|---|---|---|
| signe | exposant | mantisse |

# Ex. 2, étape 1: décimal vers binaire (IEEE754)

Convertir 12,5 en binaire sur 32 bits avec IEEE 754

- 12,5 est positif, donc bit de signe = 0

| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- $12,5 = 0b1100,1 = 1,1001 \times 2^3$

- nous voulons que exposant - 127 = 3, alors exposant = 130
  soit 0b10000010

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

- la mantisse est 1001000…

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

(signe) 1,mantisse x $2^{(exposant-127)}$

| 1 bit | 8 bits | 23 bits |
|---|---|---|
| signe | exposant | mantisse |

# Ex. 2, étape 2: binaire vers hexadécimal

Écrire votre réponse en hexadécimal

| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | | | | 1 | | | | 4 | | | | 8 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | |

| Hexadécimal | Binaire |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

donc, 0x41480000

# Considérations

- Intervalle: 1.2E-38 to 3.4E+38
  - Plus que 4G valeurs mais seulement 32 bits?
  - 126765060022822940149670320537 6 Oui
  - 126765060022822940149670320537 7 Non
- $(2^{60}-2^{60})+1$ donne 0+1 = 1 car la soustraction se fait exactement;
- $(2^{60}+1)-2^{60}$ donne 0, car, avec une précision limitée à 53 bits, $2^{60}+1$ s'arrondit à $2^{60}$.
- Même fonctionnement pour les autres tailles de nombres à virgule flottante mais avec plus de bits

# Cas spéciaux IEEE754

- Exposant et mantisse à 0:
  - 0x00000000
  - Le nombre est zéro (+0 ou -0)
- Exposant à 255 et mantisse à 0
  - 0x7F800000 ou 0xFF800000
  - $+\infty$ ou $-\infty$
- Exposant à 255 et mantisse différente de 0
  - ex: 0xFFC00000
  - Pas un nombre («not a number», ou NaN)
  - Exemples: log(-1), 0/0.

# Outils pratiques

http://www.binaryconvert.com

https://float.exposed/0x3f800000

# PHIR™ #3

- A priori, **nous ne pouvons pas savoir** ce qu'une chaîne binaire signifie.
  - Ex: que veut dire 0x416C6C6F (sur 32 bits)?
  - La bonne réponse est: ça dépend!

| | |
|---|---|
| entier non-signé | 1097624687 |
| entier signé | 1097624687 |
| réel | 14,47764 |

- Il nous faut donc savoir quel format utiliser pour bien interpréter les données

# Chaînes de caractères — ASCII

- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- Table reliant un caractère d'imprimerie à une valeur de 0x00 à 0x7F
  - donc nécessite 7 bits dans sa version originale

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

http://www.asciitable.com

# Chaînes de caractères — ASCII

- Exemple: « Bonjour! » (sans les « ») en ASCII?
  - attention aux majuscules…
- Bonjour! = 0x42 6F 6E 6A 6F 75 72 21

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Source: www.LookupTables.com

http://www.asciitable.com

# ASCII: 7 ou 8 bits?

Quelle est la différence entre
écrire la chaîne de caractères « ABC » sur **7 bits**
et l'écrire sur **8 bits**?

# ASCII: 7 ou 8 bits?

Quelle est la différence entre
écrire la chaîne de caractères « ABC » sur **7 bits**
et l'écrire sur **8 bits**?

- Sur 7 bits:
  - « A »: 0x41 = 0b1000001
  - « B »: 0x42 = 0b1000010
  - « C »: 0x43 = 0b1000011
- Placer tous les bits un à la suite de l'autre:
  - 0b10000011000010100011
- Convertir ensuite en hexadécimal en regroupant par groupe de 4:
  - 0b 1 0000 0110 0001 0100 0011
  - 0x 1 0 6 1 4 3
- donc, 0x106143.

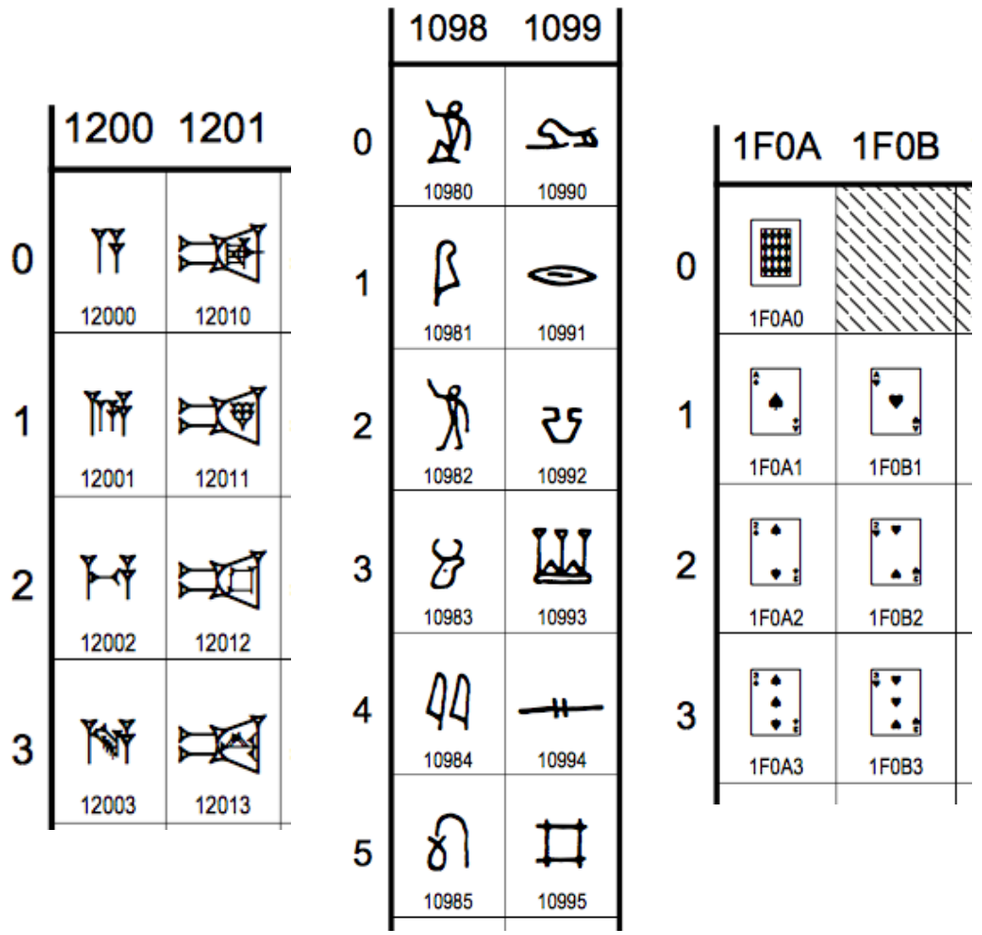
# ASCII: 7 ou 8 bits?

Quelle est la différence entre
écrire la chaîne de caractères « ABC » sur **7 bits**
et l'écrire sur **8 bits**?

- Sur 8 bits
  - « A »: 0x41 = 0b01000001
  - « B »: 0x42 = 0b01000010
  - « C »: 0x43 = 0b01000011
- Placer tous les bits un à la suite de l'autre:
  - 0b010000010100001001000011
- Convertir ensuite en hexadécimal en regroupant par groupe de 4:
  - 0b 0100 0001 0100 0010 0100 0011
  - 0x 4 1 4 2 4 3
- donc, 0x414243.

# Chaînes de caractères — Unicode

- ASCII ne suffit pas à représenter tous les caractères
- Standard visant à attribuer un numéro distinct à chaque caractère
- Couvre même
  - l'écriture cunéiforme
  - hiéroglyphes
  - cartes à jouer



# UTF-8 (Unicode Transformation Format — 8 bits)

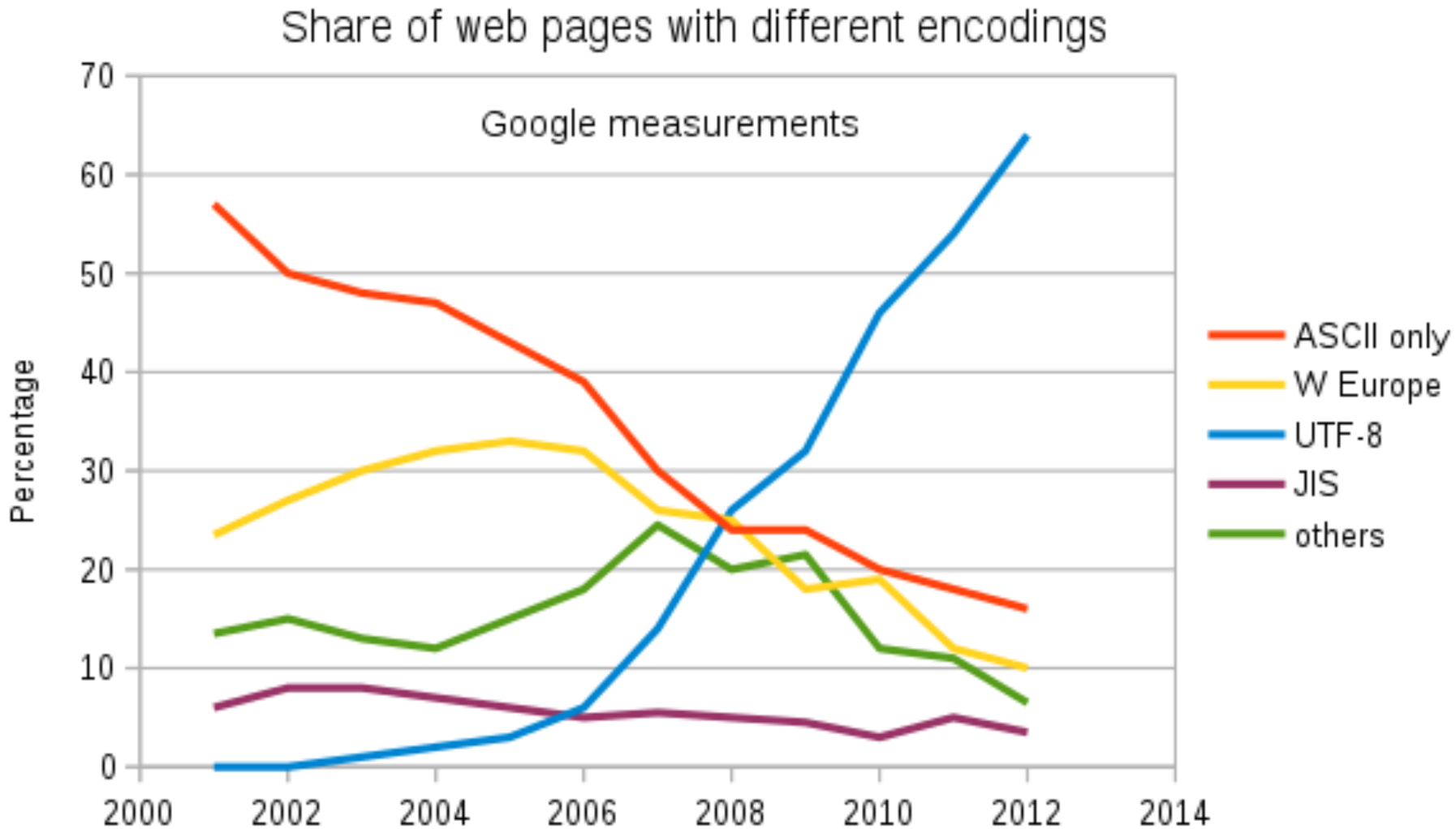
- Système de préfixe par groupes de 8 bits (1 octet)
  - Si premier bit (MSB) est 0, alors 1 seul octet
  - Sinon:

Rétrocompatibilité avec ASCII

**Définition du nombre d'octets utilisés dans le codage (uniquement les séquences valides)**

| Caractères codés | Représentation binaire UTF-8 | Premier octet valide (hexadécimal) | Signification |
|---|---|---|---|
| **U+0000 à U+007F** | 0xxxxxxx | 00 à 7F | 1 octet codant 1 à 7 bits |
| **U+0080 à U+07FF** | 110xxxxx 10xxxxxx | C2 à DF | 2 octets codant 8 à 11 bits |
| **U+0800 à U+0FFF** | 11100000 101xxxxx 10xxxxxx | E0 (le 2e octet est restreint de A0 à BF) | 3 octets codant 12 à 16 bits |
| **U+1000 à U+1FFF** | 11100001 10xxxxxx 10xxxxxx | E1 | |
| **U+2000 à U+3FFF** | 1110001x 10xxxxxx 10xxxxxx | E2 à E3 | |
| **U+4000 à U+7FFF** | 111001xx 10xxxxxx 10xxxxxx | E4 à E7 | |
| **U+8000 à U+BFFF** | 111010xx 10xxxxxx 10xxxxxx | E8 à EB | |
| **U+C000 à U+CFFF** | 11101100 10xxxxxx 10xxxxxx | EC | |
| **U+D000 à U+D7FF** | 11101101 100xxxxx 10xxxxxx | ED (le 2e octet est restreint de 80 à 9F) | |
| **U+E000 à U+FFFF** | 1110111x 10xxxxxx 10xxxxxx | EE à EF | |
| **U+10000 à U+1FFFF** | 11110000 1001xxxx 10xxxxxx 10xxxxxx | F0 (le 2e octet est restreint de 90 à BF) | 4 octets codant 17 à 21 bits |
| **U+20000 à U+3FFFF** | 11110000 101xxxxx 10xxxxxx 10xxxxxx | | |
| **U+40000 à U+7FFFF** | 11110001 10xxxxxx 10xxxxxx 10xxxxxx | F1 | |
| **U+80000 à U+FFFFF** | 1111001x 10xxxxxx 10xxxxxx 10xxxxxx | F2 à F3 | |
| **U+100000 à U+10FFFF** | 11110100 1000xxxx 10xxxxxx 10xxxxxx | F4 (le 2e octet est restreint de 80 à 8F) | |

# Popularité






Source: https://en.wikipedia.org/wiki/UTF-8

# PHIR™ #3

- A priori, **nous ne pouvons pas savoir** ce qu'une chaîne binaire signifie.
  - Ex: que veut dire 0x416C6C6F (sur 32 bits)?
  - La bonne réponse est: ça dépend!

| | |
|---|---|
| entier non-signé | 1097624687 |
| entier signé | 1097624687 |
| réel | 14.47764 |
| caractères ASCII | Allo |

- Il nous faut donc savoir **quel format** (quelle « recette ») utiliser pour bien interpréter les données

# En résumé: 4 PHIRs™*

| Raccourci | Explication |
|---|---|
| tout en binaire | Dans un ordinateur, tout, absolument tout, est stocké en format binaire. |
| # de bits prédéterminé | On utilise un nombre fini et pré-déterminé de bits pour représenter de l'information. |
| besoin d'une recette | À priori, nous ne pouvons savoir ce qu'une chaîne binaire signifie, il nous faut une «recette». |
| hexadécimal = binaire | L'hexadécimal est une façon plus compacte de représenter du binaire. |



*PHIR: Points Hyper Importants à Retenir, ou ce qui arrive quand le prof essaie d'être drôle. Désolé.