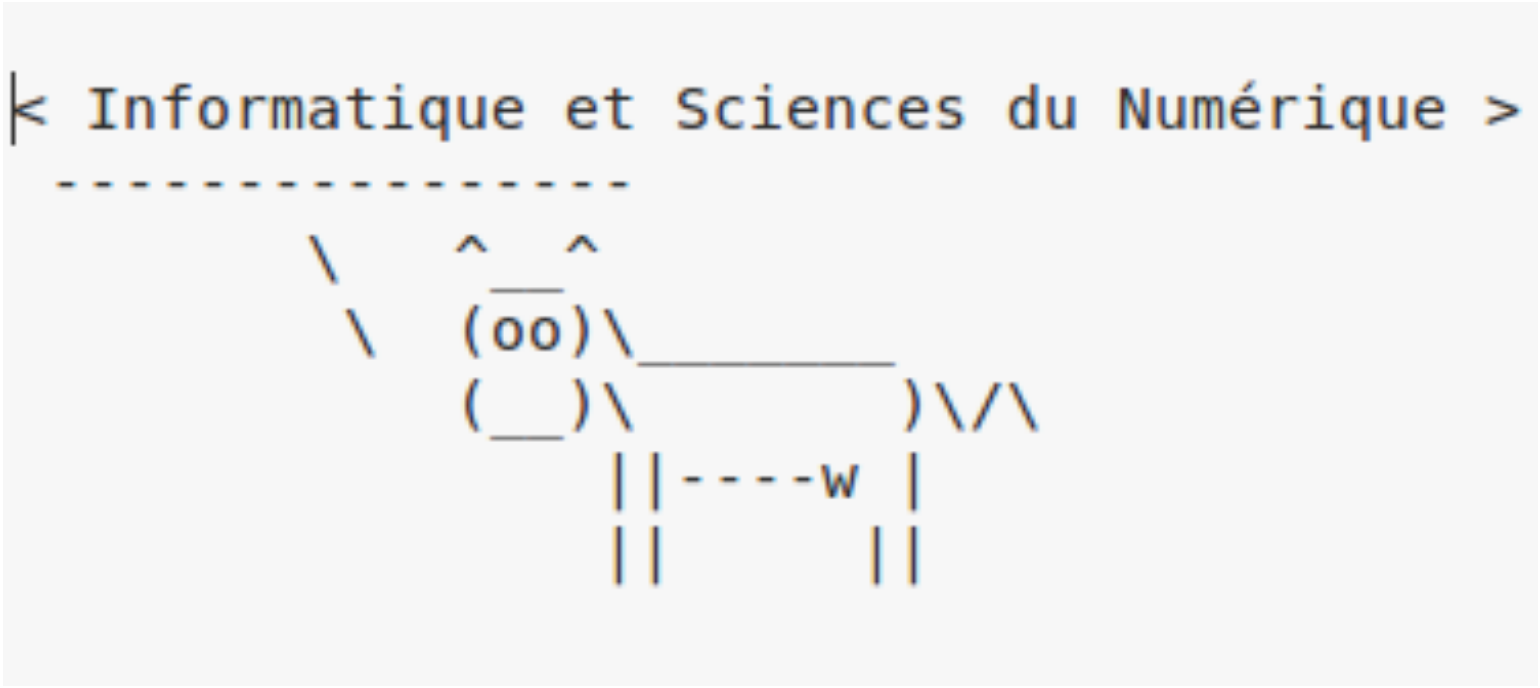
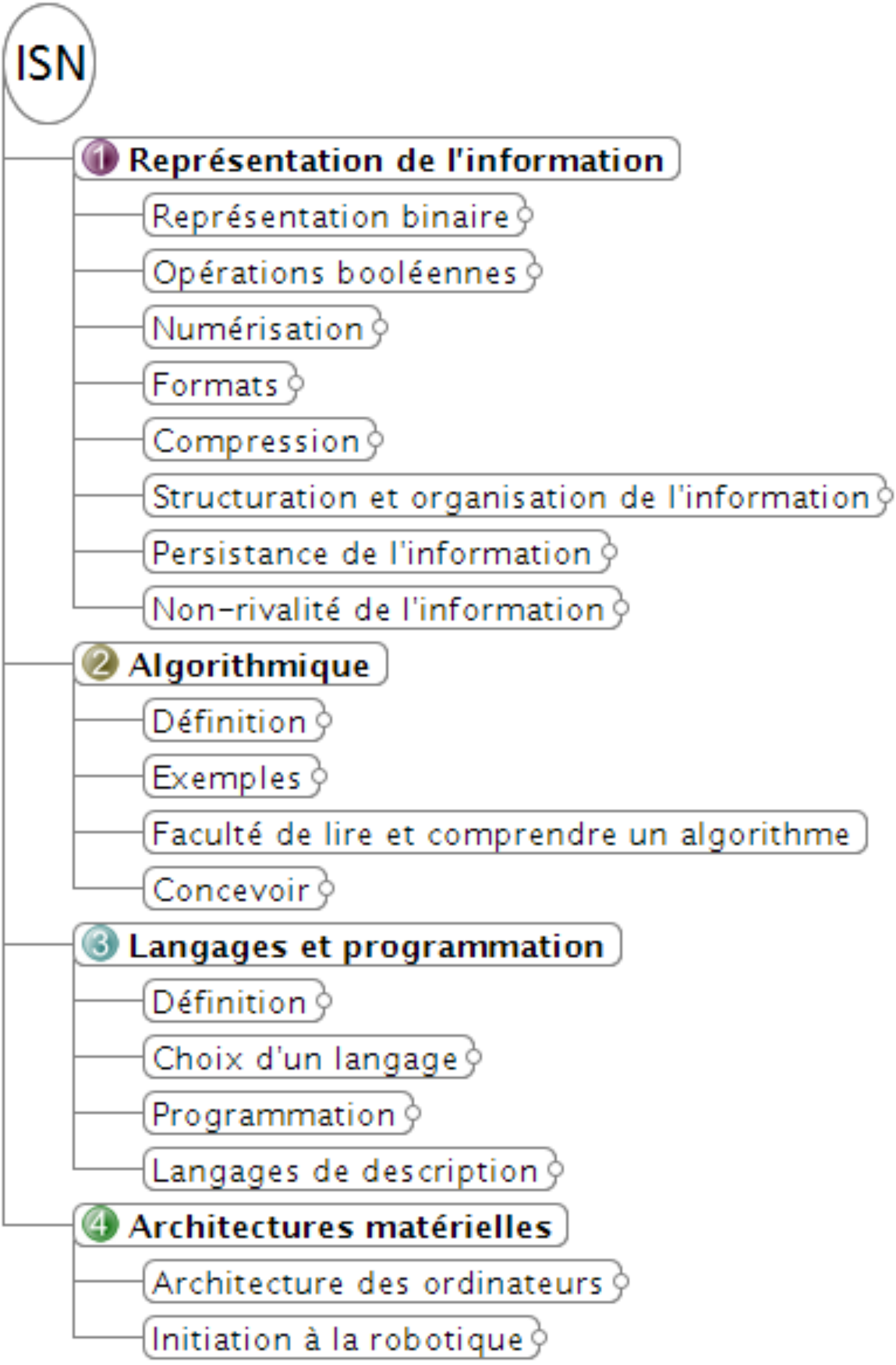
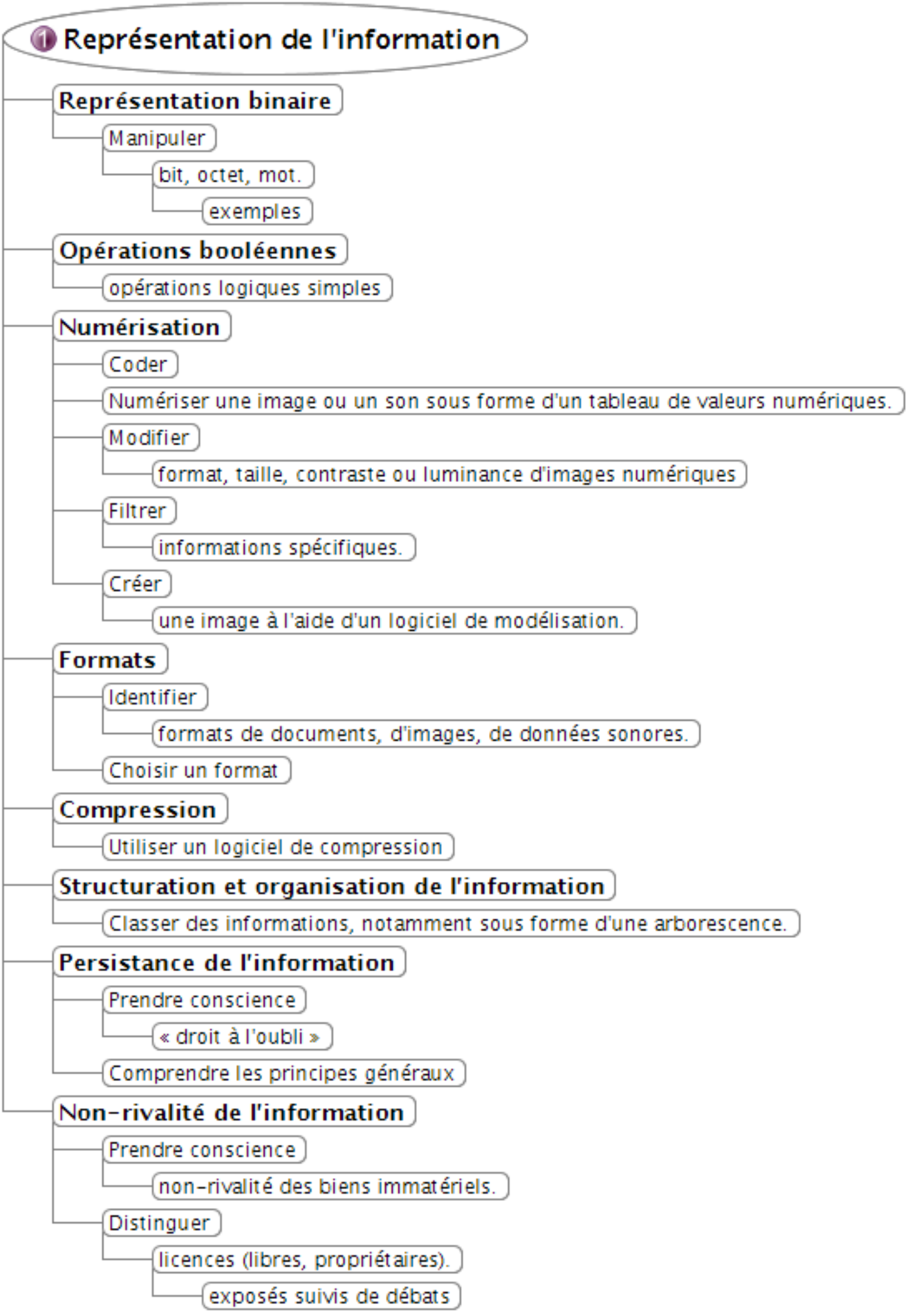
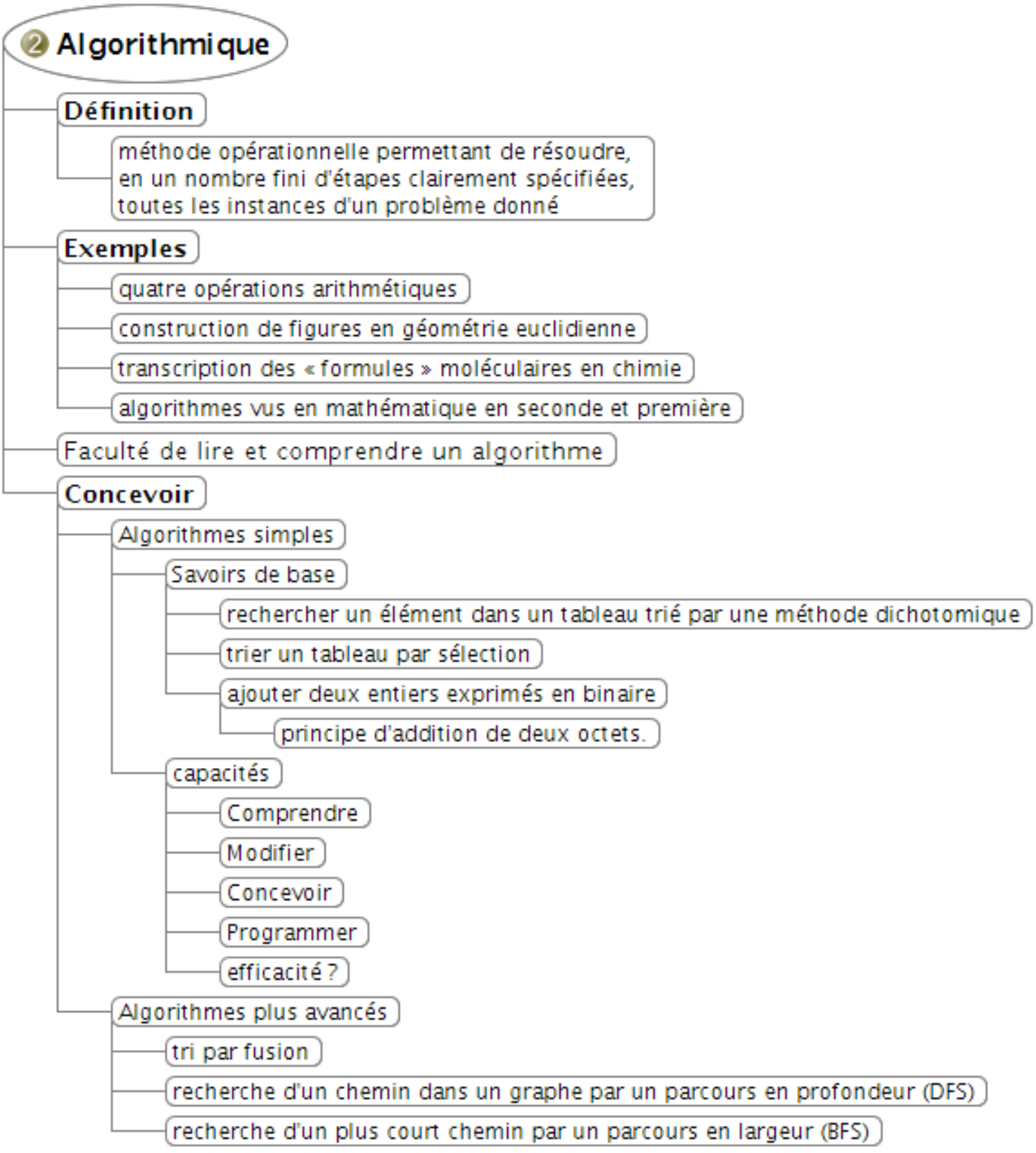
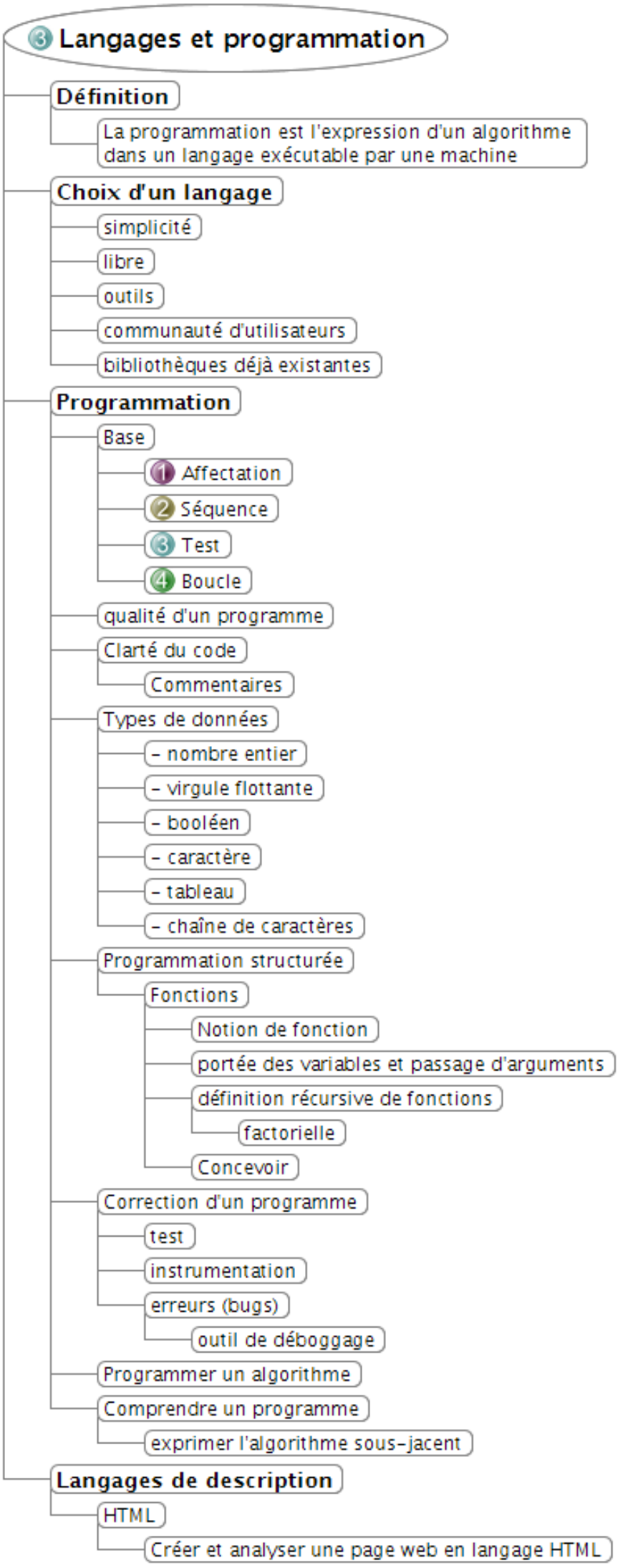
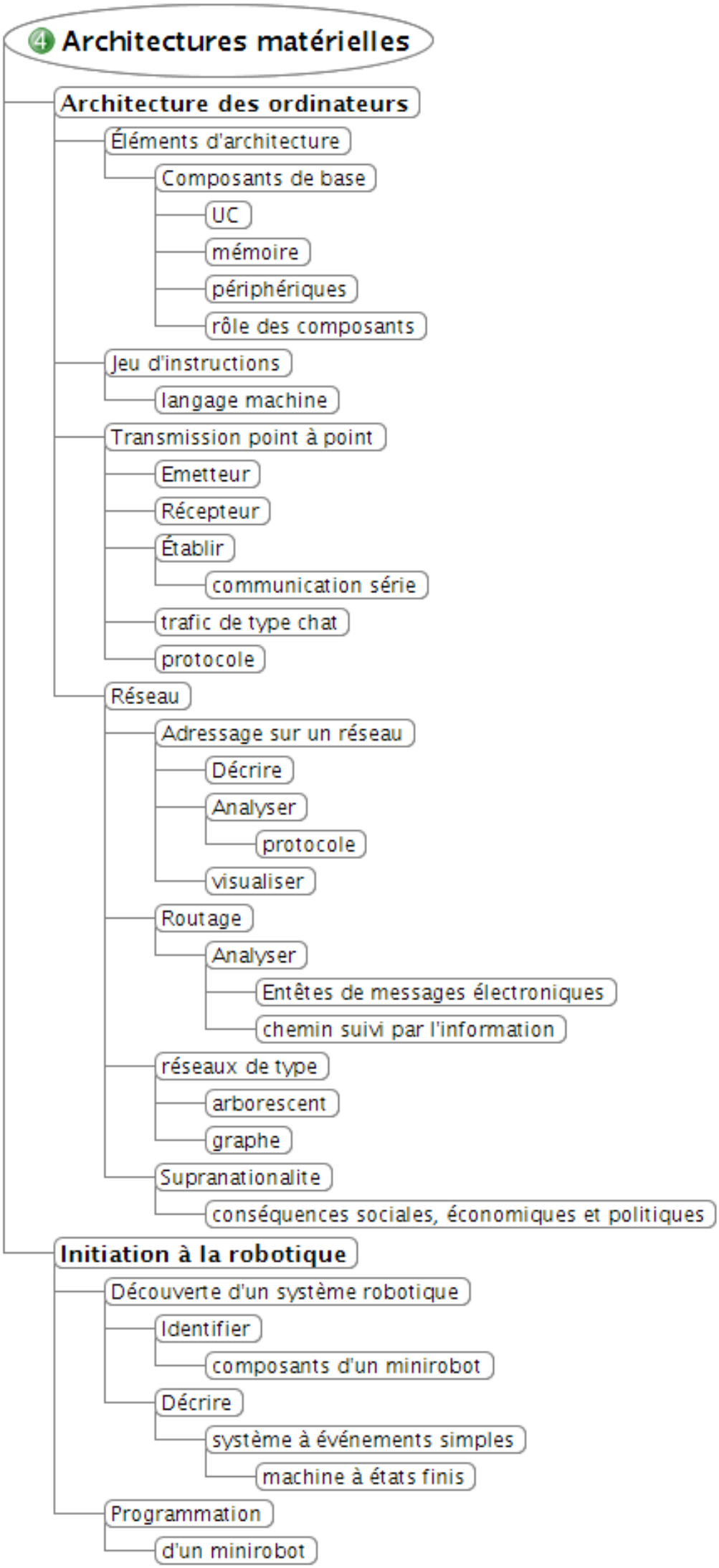

< Informatique et Sciences du Numérique >
\ ^__^
\ (oo)_____
(__)\ )\/\
||----w |
|| ||


ISN
1 Représentation de l'information
Représentation binaire
Opérations booléennes
Numérisation
Formats
Compression
Structuration et organisation de l'information
Persistance de l'information
Non-rivalité de l'information
2 Algorithmique
Définition
Exemples
Faculté de lire et comprendre un algorithme
Concevoir
3 Langages et programmation
Définition
Choix d'un langage
Programmation
Langages de description
4 Architectures matérielles
Architecture des ordinateurs
Initiation à la robotique


1 Représentation de l'information
Représentation binaire
Manipuler
bit, octet, mot.
exemples
Opérations booléennes
opérations logiques simples
Numérisation
Coder
Numériser une image ou un son sous forme d'un tableau de valeurs numériques.
Modifier
format, taille, contraste ou luminance d'images numériques
Filtrer
informations spécifiques.
Créer
une image à l'aide d'un logiciel de modélisation.
Formats
Identifier
formats de documents, d'images, de données sonores.
Choisir un format
Compression
Utiliser un logiciel de compression
Structuration et organisation de l'information
Classer des informations, notamment sous forme d'une arborescence.
Persistance de l'information
Prendre conscience
« droit à l'oubli »
Comprendre les principes généraux
Non-rivalité de l'information
Prendre conscience
non-rivalité des biens immatériels.
Distinguer
licences (libres, propriétaires).
exposés suivis de débats


2 Algorithmique
Définition
méthode opérationnelle permettant de résoudre, en un nombre fini d'étapes clairement spécifiées, toutes les instances d'un problème donné
Exemples
quatre opérations arithmétiques
construction de figures en géométrie euclidienne
transcription des « formules » moléculaires en chimie
algorithmes vus en mathématique en seconde et première
Faculté de lire et comprendre un algorithme
Concevoir
Algorithmes simples
Savoirs de base
rechercher un élément dans un tableau trié par une méthode dichotomique
trier un tableau par sélection
ajouter deux entiers exprimés en binaire
principe d'addition de deux octets.
capacités
Comprendre
Modifier
Concevoir
Programmer
efficacité ?
Algorithmes plus avancés
tri par fusion
recherche d'un chemin dans un graphe par un parcours en profondeur (DFS)
recherche d'un plus court chemin par un parcours en largeur (BFS)


3 Langages et programmation
Définition
La programmation est l'expression d'un algorithme dans un langage exécutable par une machine
Choix d'un langage
simplicité
libre
outils
communauté d'utilisateurs
bibliothèques déjà existantes
Programmation
Base
1 Affectation
2 Séquence
3 Test
4 Boucle
qualité d'un programme
Clarté du code
Commentaires
Types de données
- nombre entier
- virgule flottante
- booléen
- caractère
- tableau
- chaîne de caractères
Programmation structurée
Fonctions
Notion de fonction
portée des variables et passage d'arguments
définition récursive de fonctions
factorielle
Concevoir
Correction d'un programme
test
instrumentation
erreurs (bugs)
outil de déboggage
Programmer un algorithme
Comprendre un programme
exprimer l'algorithme sous-jacent
Langages de description
HTML
Créer et analyser une page web en langage HTML


4 Architectures matérielles
Architecture des ordinateurs
Éléments d'architecture
Composants de base
UC
mémoire
périphériques
rôle des composants
Jeu d'instructions
langage machine
Transmission point à point
Emetteur
Récepteur
Établir
communication série
trafic de type chat
protocole
Réseau
Adressage sur un réseau
Décrire
Analyser
protocole
visualiser
Routage
Analyser
Entêtes de messages électroniques
chemin suivi par l'information
réseaux de type
arborescent
graphe
Supranationalite
conséquences sociales, économiques et politiques
Initiation à la robotique
Découverte d'un système robotique
Identifier
composants d'un minirobot
Décrire
système à événements simples
machine à états finis
Programmation
d'un minirobot

DES PUISSANCES DE 2 AU CODE ANDROMEDE

--> préliminaire : Les puissances de 2

Ecrire les 11 premières puissances de 2.
Il serait bien de connaître par coeur les 11 premières puissance de 2
1024 = ?

exercice python :
- écrire un programme qui affiche les puissances de 2 de 2^0 à 2^20
- écrire un programme qui, connaissant une puissance de 2, renvoie la puissance (pour m=2^n, l'entrée c'est m, la sortie c'est n)

correction partielle code python :

```
>>> for i in range(0,20):
...     print 2**i
...
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
```

--> système binaire

Dans notre système pour compter, dit décimal, on a :
1243 = 1000 + 200 + 40 + 3
soit :
1243 = 10^3 + 2*10^2 + 4*10^1 + 3*10^0

Nous sommes en base 10. On utilise alors les 10 symboles 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 pour noter les nombres.
Dans le système binaire, en base 2, nous aurons 2 symboles 0 et 1 pour noter les nombres.

Ainsi, tout nombre en binaire pourra s'écrire :
 a = a_n*2^n + … + a_2*2^2 + a_1*2^1 + a_0*2^0
 où les a_i appartiennent à l'ensemble {0,1}

 Par exemple
     1001 = 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0
     Ainsi, b1001 = 9

vérification python :

```
>>> bin(1001)
>>> '9'
```

décimal binaire
0 0
1 1
2 10
3 11
4 100
5 101

exercice python :
- écrire un programme qui affiche en binaire les entiers naturels de 1 à 20

décimal vers binaire

on effectue les divisions entières successives par 2 du nombre à transformer.
127 = 63*2 + 1
63 = 31*2 + 1
31 = 15*2 + 1
15 = 7*2 + 1
7 = 3*2 + 1
3 = 1*2 + 1

127 = (((((1*2 + 1)*2 + 1)*2 + 1)*2 + 1)*2 + 1)*2 + 1
127 = 1*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0

D'où 127 s'écrit 1111111 en binaire.

vérification python :
>>> bin(1111111)
>>> '127'

exercice python :
- écrire un programme qui convertit en binaire un entier naturel.

-> un mot sur l'hexadécimal

Le système hexadécimal correspond à la base 16. On le rencontre entre autres dans le codage des couleurs.
On utilise les 16 symboles suivant pour représenter les nombres :
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

A vaut 10, B vaut 11, etc..

Ainsi FF = 15*16^1 + 15*16^0 = 255

Pour coder 127 en hexadécimal, il faut effectuer les divisions successives par 16 et procéder comme pour le binaire.
On peut s'aider de python pour le calcul
>>> 127/16
>>> 7
>>> 127%16
>>> 15

Donc 127 = 7*16 + 15

127 = 7*16^1 + 15*16^0
127 s'écrit donc 7F (parce que 15 c'est le symbole F)

exercice python :
- écrire un programme qui convertit en hexadécimal un entier naturel, et vice-versa.

--> Codage d'un nombre

un bit c'est 0 ou 1

un octet c'est 8 bits

Le plus grand nombre entier naturel que l'on peut coder sur 8 bits est donc :
11111111

code python :
>>> bin(11111111)
>>> '255'

Quel est le plus grand entier que l'on peut coder sur 16 bits ?

En remarquant que :
>>> bin(65535)
'0b1111111111111111'
>>>

il faudra donc 16 bits pour coder 65535.

--> Codage d'un texte

On utilise le code ASCII (wikipedia) qui donne en hexadécimal le code ASCII des caractères du clavier.
Table fournie : ASCII_Code_Chart.svg

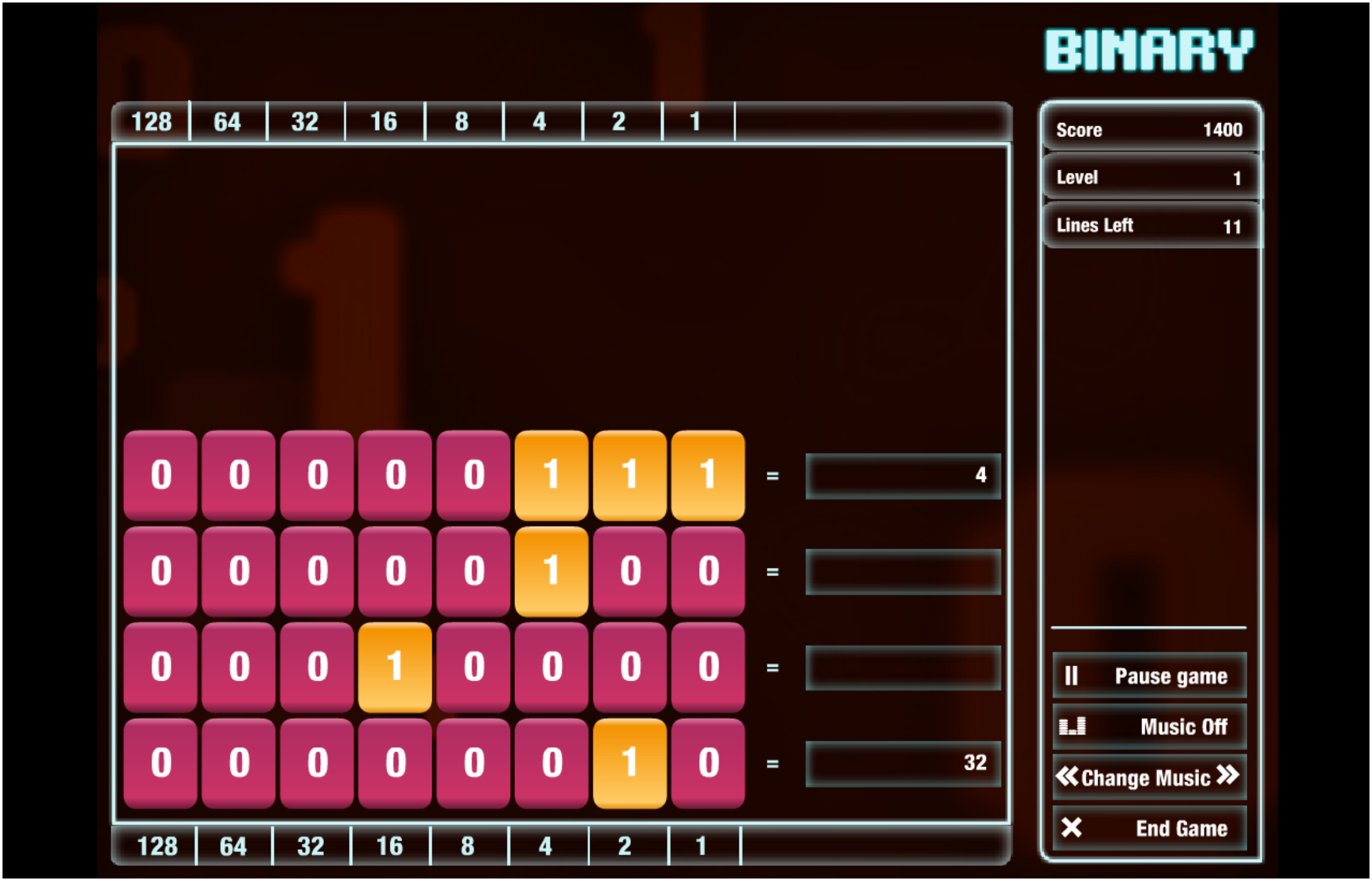
Exemple : A = 41 en hexadécimal = 65 en décimal
Pour coder le caractère A, on va donc coder en binaire le nombre 65 : 1000001
Il suffit de 7 bits pour coder les caractères de l'alphabet. On utilise le 8ème bit pour ...

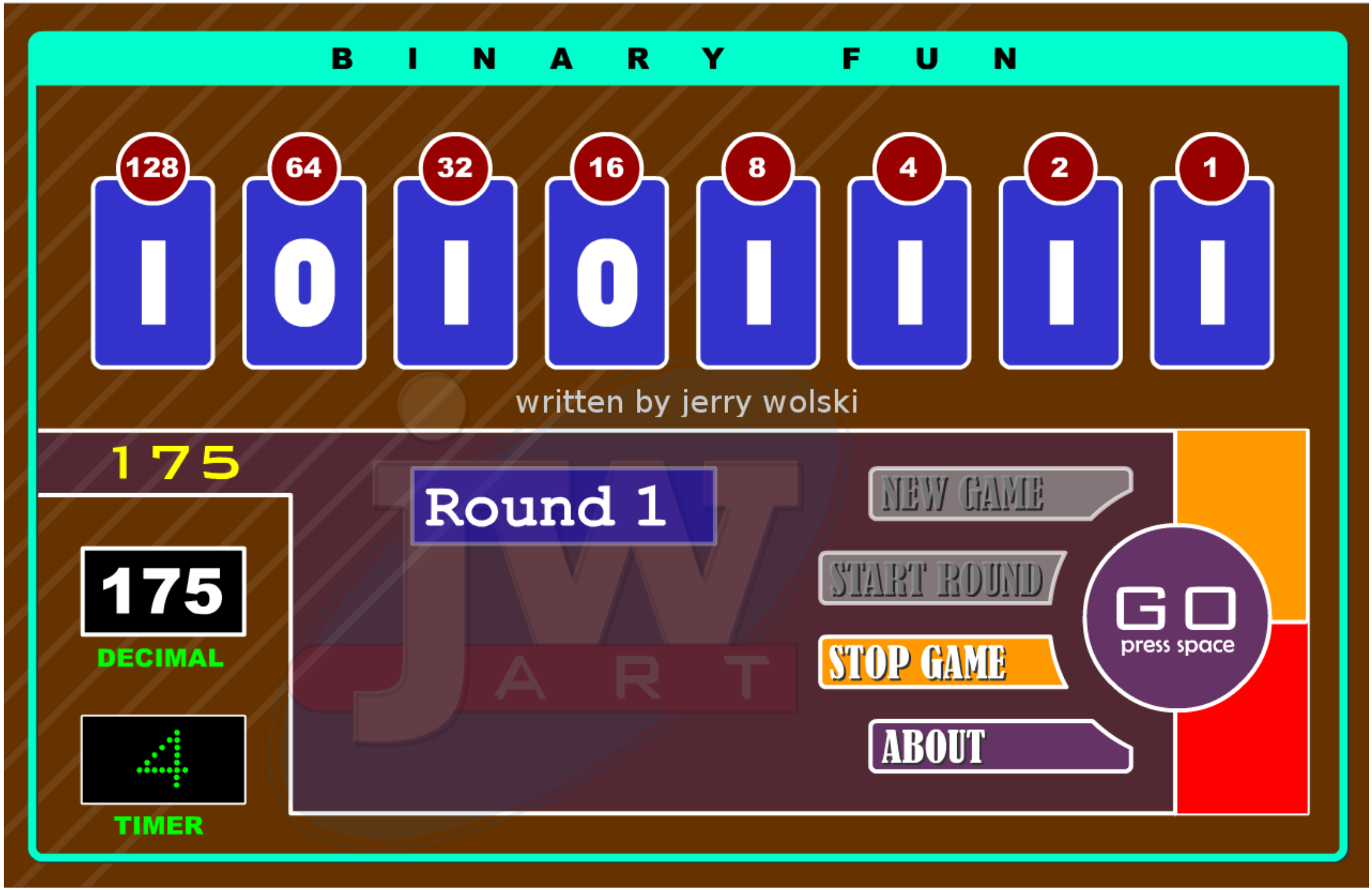
Décoder en français la suite binaire suivante pour connaître le message envoyé du futur dans le film :
The.Andromeda.Strain. (2008 American science-fiction film)
basée sur une nouvelle de Michael Crichton (1969) (http://en.wikipedia.org/wiki/The_Andromeda_Strain).

0110111 0110011 0111001 0110101 0110010 0111000
1000010 1000011 1000001 1001001 1001100 1001100
1010101 1010011 1001001 1001110 1000110 1000101
1010010 1001110 1010101 1010011

Voir l'extrait de film pour connaître le message.


BINARY
128 64 32 16 8 4 2 1
Score 100
Level 1
Lines Left 14
7 8 9
4 5 6
1 2 3
DEL 0
0 0 0 0 0 0 1 0
=
2
128 64 32 16 8 4 2 1
Pause game
Music Off
Change Music
End Game


BINARY
128 64 32 16 8 4 2 1
0 0 0 0 0 1 1 1 = 4
0 0 0 0 0 1 0 0 =
0 0 0 1 0 0 0 0 =
0 0 0 0 0 0 1 0 = 32
128 64 32 16 8 4 2 1
Score 1400
Level 1
Lines Left 11
Pause game
Music Off
Change Music
End Game

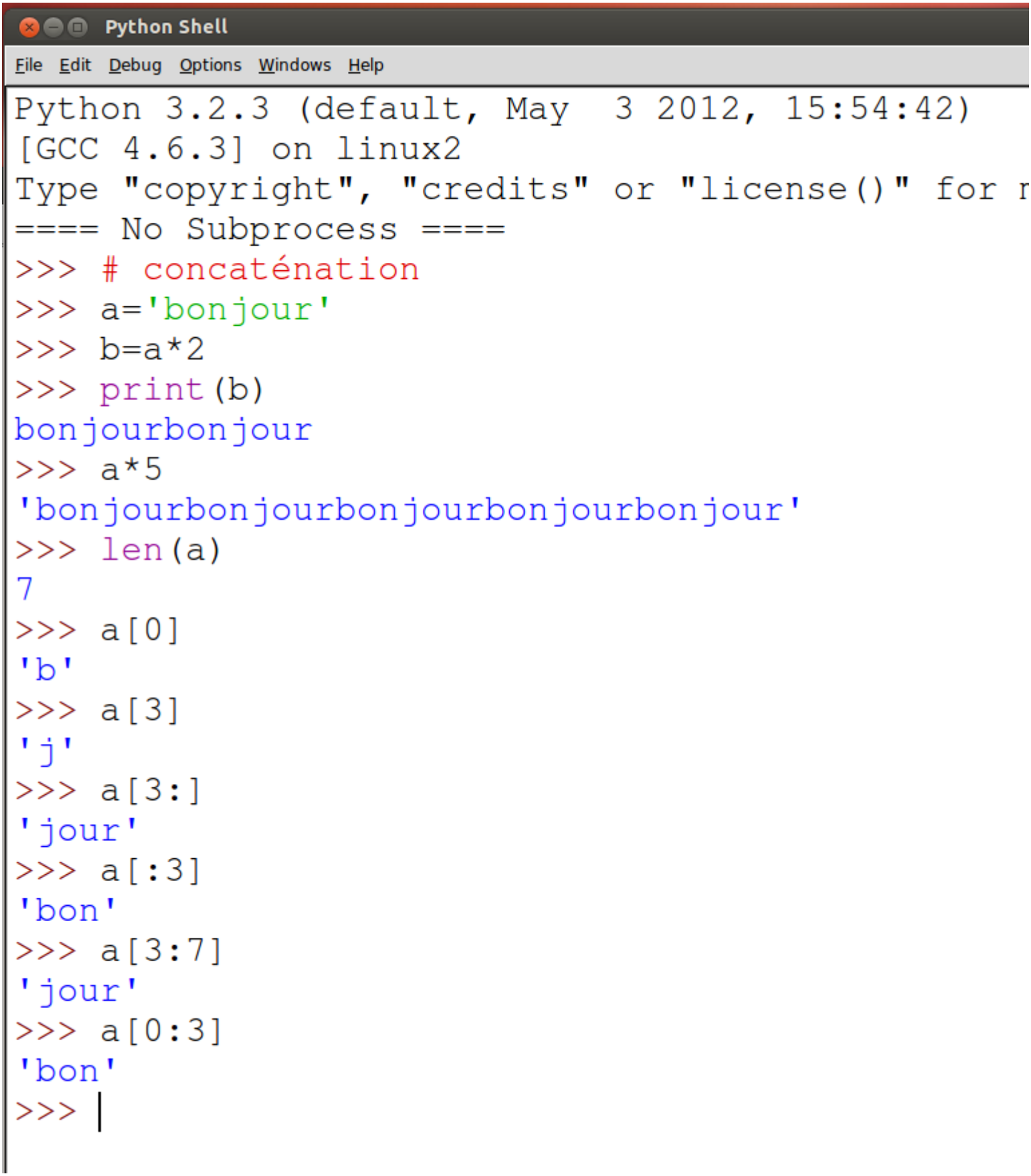

B I N A R Y F U N
128
64
32
16
8
4
2
1
I
0
I
0
I
I
I
I
written by jerry wolski
175
Round 1
NEW GAME
START ROUND
STOP GAME
ABOUT
GO
press space
175
DECIMAL
4
TIMER

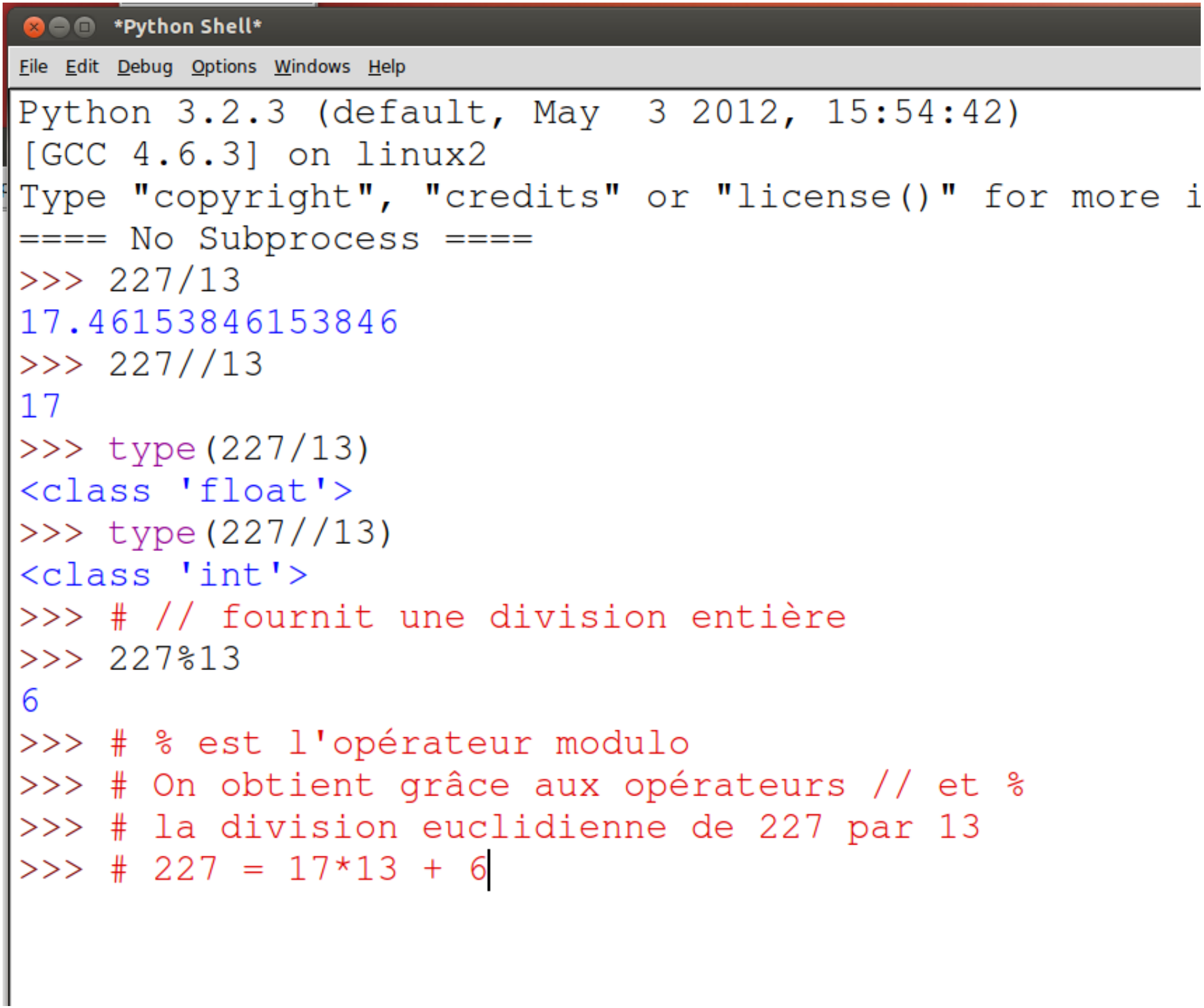
$$0110111_2 = (32+16+4+2+1)_{10} = 55_{10}$$

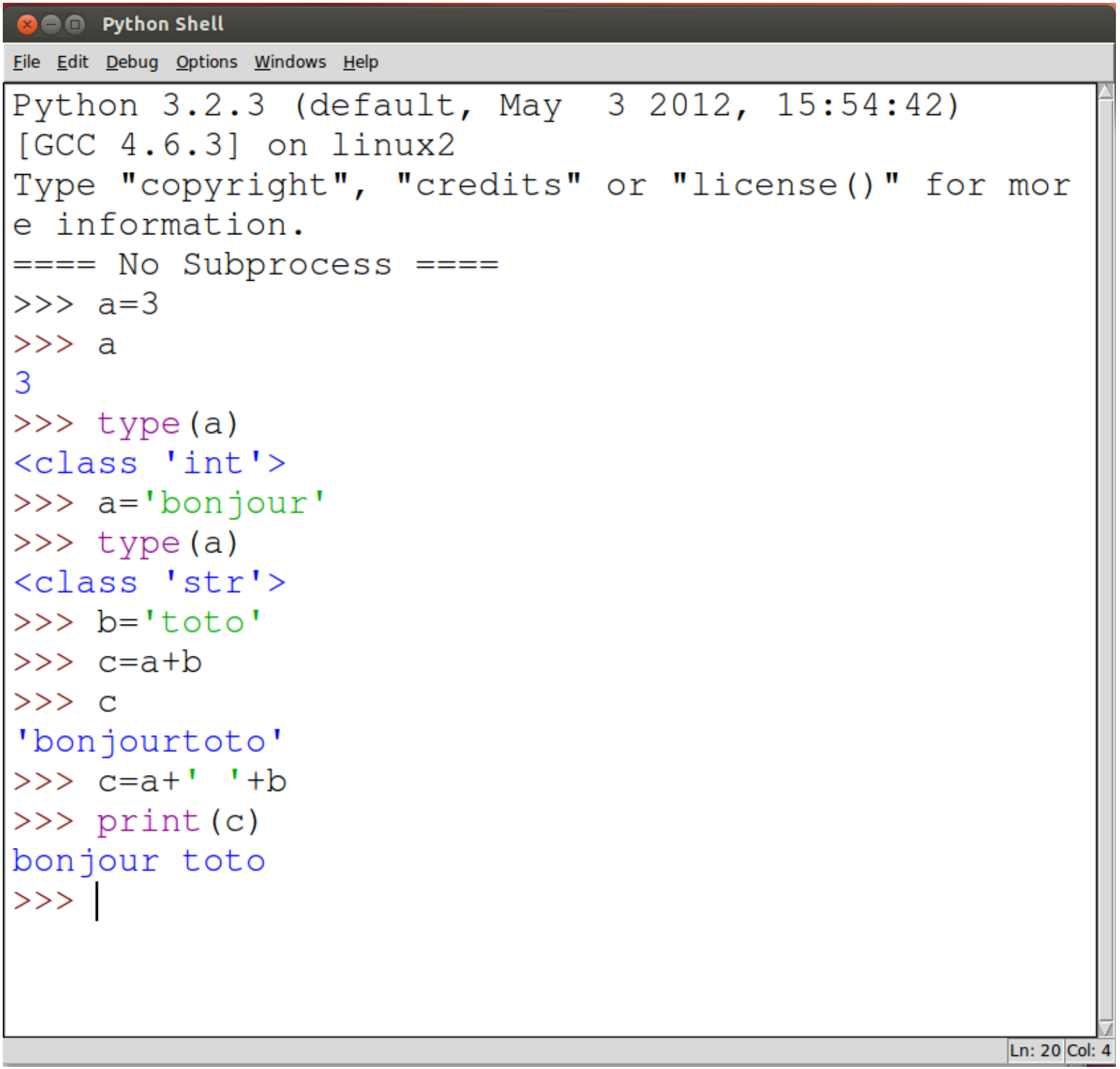
$$55_{10} = 37_{16}$$

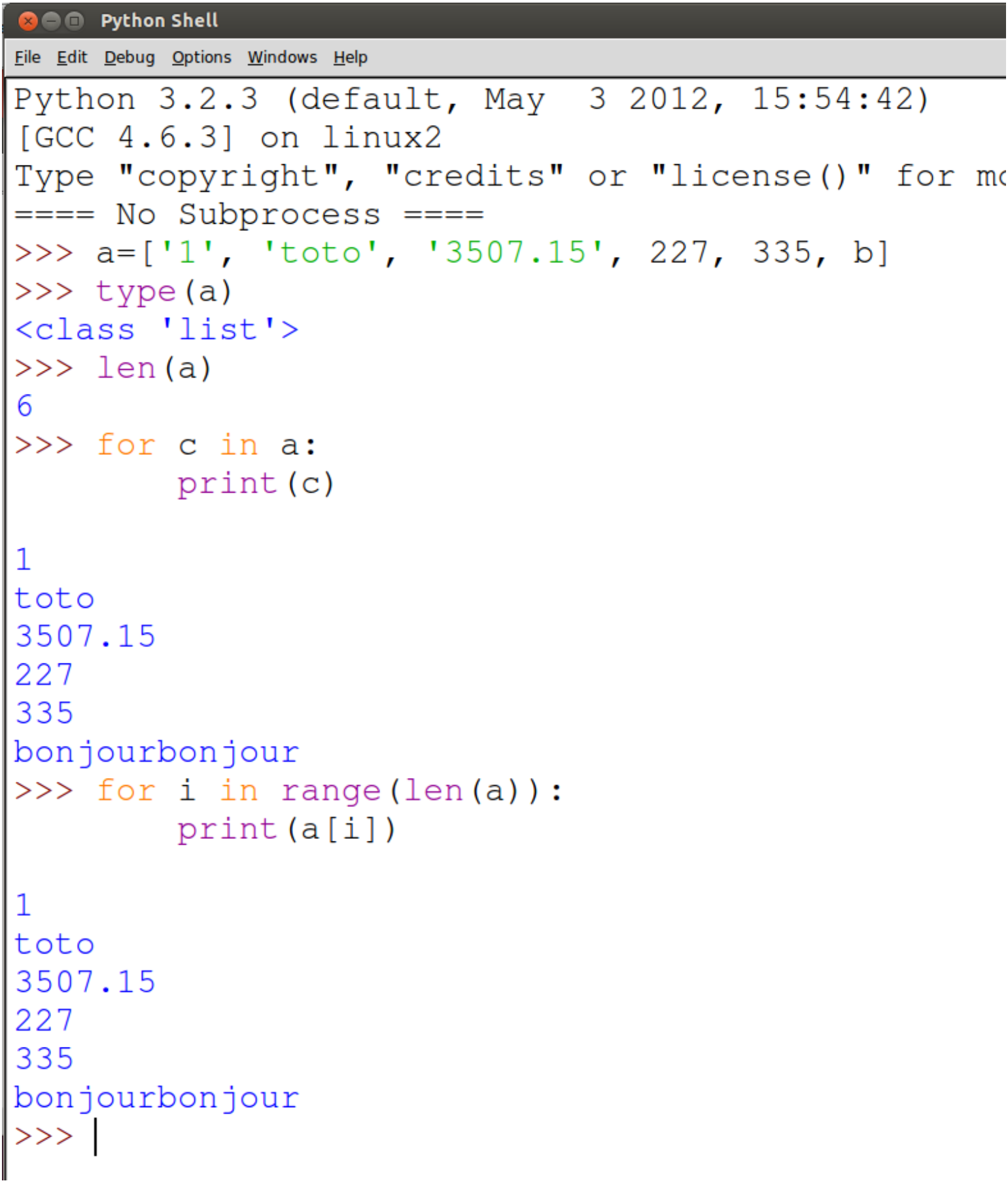
car $55 = 3 \times 16 + 7$

$37_{16} = 7$ dans le code ASCII


Python Shell
File Edit Debug Options Windows Help
Python 3.2.3 (default, May 3 2012, 15:54:42)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for r
==== No Subprocess ====
>>> # concaténation
>>> a='bonjour'
>>> b=a*2
>>> print(b)
bonjourbonjour
>>> a*5
'bonjourbonjourbonjourbonjourbonjour'
>>> len(a)
7
>>> a[0]
'b'
>>> a[3]
'j'
>>> a[3:]
'jour'
>>> a[:3]
'bon'
>>> a[3:7]
'jour'
>>> a[0:3]
'bon'
>>>


Python Shell
File Edit Debug Options Windows Help
Python 3.2.3 (default, May 3 2012, 15:54:42)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more i
==== No Subprocess ====
>>> 227/13
17.46153846153846
>>> 227//13
17
>>> type(227/13)
<class 'float'>
>>> type(227//13)
<class 'int'>
>>> # // fournit une division entière
>>> 227%13
6
>>> # % est l'opérateur modulo
>>> # On obtient grâce aux opérateurs // et %
>>> # la division euclidienne de 227 par 13
>>> # 227 = 17*13 + 6


Python Shell
File Edit Debug Options Windows Help
Python 3.2.3 (default, May 3 2012, 15:54:42)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for mor
e information.
==== No Subprocess ====
>>> a=3
>>> a
3
>>> type(a)
<class 'int'>
>>> a='bonjour'
>>> type(a)
<class 'str'>
>>> b='toto'
>>> c=a+b
>>> c
'bonjourtoto'
>>> c=a+' '+b
>>> print(c)
bonjour toto
>>>
Ln: 20 Col: 4


Python Shell
File Edit Debug Options Windows Help
Python 3.2.3 (default, May 3 2012, 15:54:42)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for mo
==== No Subprocess ====
>>> a=['1', 'toto', '3507.15', 227, 335, b]
>>> type(a)
<class 'list'>
>>> len(a)
6
>>> for c in a:
print(c)
1
toto
3507.15
227
335
bonjourbonjour
>>> for i in range(len(a)):
print(a[i])
1
toto
3507.15
227
335
bonjourbonjour
>>>

```
A=[True,False]
B=[True,False]
for a in A:
    for b in B:
       print (' A = ',a,' B= ', b, 'A et B = ',a and b)
       print (' A = ',a,' B= ', b, 'A ou B = ',a or b)
```

Logique avec Python

--> Qu'est-ce qu'un booléen ?

http://fr.wikipedia.org/wiki/Bool%C3%A9en
C'est une variable qui ne peux prendre que deux valeurs : VRAI ou FAUX.
En Python, le type d'une telle variable est bool, les deux valeurs possibles sont True ou False.

--> Expressions booléennes

Une expression booléenne a deux valeurs possibles : False ou True.
Python attribut à une expression booléenne la valeur False si c'est :
• la constante False
• la constante None
• une séquence ou une collection vide
• une donnée numérique de valeur 0. Tout le reste vaut ???.

code Python

```
>>> type(False)
<class 'bool'>
>>> type(True)
<class 'bool'>
>>> False
False
>>> bool(None)
False
>>> bool(' ')
True
>>> bool('')
False
>>> bool(0)
False
>>> bool(156.87)
True
```

--> Opérateurs relationnels ou de comparaison :

Ce sont les opérateurs ==, !=, >, >=, < et <=

* Illustration pour x=7 et y=17

Opérateur	Expression	Signification	Valeur
==	x == y	Égal	0 (faux)
!=	x != y	Non égal	1 (vrai)
>	x> y	Plus grand que	0
<	x < y	Plus petit que	1
>=	x >= y	Plus grand ou égal à	0
<=	x <= y	Plus petit ou égal à	1
is	x is y	est le même objet	0
is not	x is not y	n'est pas le même objet	1

code Python

```
>>> x=7
>>> y=17
>>> x==y
False
>>> x!=y
True
>>> x>y
False
>>> x>=y
False
>>> x<y
True
>>> x<=y
True
>>> x is y
False
>>> x is not y
True
```

* Illustration avec deux chaînes de caractères

code Python

```
>>>a='encyclopédie1'
>>>b='encyclopédie2'
>>> a==b
False
>>> len(a)
13
>>> a[:12]==b[:12]
True
```

Exercices Python :

1/ Ecrire un programme dont la sortie dans le shell de Python est la suivante :

_____

```
>>>Entrez un entier naturel : 45
>>>Entrez un second entier naturel : 215
>>>
>>>    x==y a pour type <class 'bool'>
>>>
>>>    x==y est un booléen : il est soit vrai, soit faux
>>>
>>>x est différent de y donc la valeur du booléen est : False
```

_____

2/ Ecrire un programme qui dira si
- un nombre x appartient à l'intervalle [a, b]
- un nombre x appartient à l'intervalle]a, b[U [c, d]
- un nombre x appartient à l'intervalle [a, b[inter]c, d]

3/
Ecrire un programme qui affiche "Bon anniversaire !" si nous sommes à la date de votre anniversaire, "Bonne journée" sinon.

--> les 3 opérateurs logiques

http://fr.wikibooks.org/wiki/Programmation_Python/Op%C3%A9rateur

Les expressions avec un opérateur logique sont évaluées à True ou False

*Le NON (négation, contraire)
p étant un booléen,
not p = 1 - p

code Python

```
>>> 2<1
False
>>> not 2<1
True
```

Table de vérité
http://fr.wikipedia.org/wiki/Table_de_v%C3%A9rit%C3%A9

p	NON p
VRAI	FAUX
FAUX	VRAI

*Le OU logique (disjonction)
http://fr.wikipedia.org/wiki/Disjonction_logique
p et q étant deux booléens,

```
    p or q
```

vaut True si p vaut True. Si p est False, l'expression est évaluée à la valeur booléenne de q.
(si p est faux, retourne q, sinon retourne p)

code Python

```
>>> 8<9
True
>>> 2<1
False
```

```
>>> (8 < 9) or (2 < 1)
True

La table de vérité d'une disjonction est donnée par le tableau suivant :
-----------------------------
|    p   |    q   || p OU q |
-----------------------------
|  VRAI  |  VRAI  ||  VRAI  |
|  VRAI  |  FAUX  ||  VRAI  |
|  FAUX  |  VRAI  ||  VRAI  |
|  FAUX  |  FAUX  ||  FAUX  |
-----------------------------

*Le ET logique (conjonction)
http://fr.wikipedia.org/wiki/Conjonction_logique
    p and q
vaut False si p est False. Si p est True, l'expression est évaluée à la valeur booléenne de q.
(si p est faux, retourne p, sinon retourne q)

code Python  :
>>> 8<9
True
>>> 2<1
False
>>> (8 < 9) and (2 < 1)
False
>>> x=36
>>> (x > 13) and (x < 27)
False
>>> x=20
>>> (x > 13) and (x < 27)
True


La table de vérité d'une conjonction est donnée par le tableau suivant :
-----------------------------
|    p   |    q   || p ET q |
-----------------------------
|  VRAI  |  VRAI  ||  VRAI  |
|  VRAI  |  FAUX  ||  FAUX  |
|  FAUX  |  VRAI  ||  FAUX  |
|  FAUX  |  FAUX  ||  FAUX  |
-----------------------------


--> Aller plus loin en logique : implication, équivalence et ou exclusif

* Implication logique
http://fr.wikipedia.org/wiki/Implication_%28logique%29
non p ou q
- Dresser à l'aide de Python la table de vérité de l'implication logique.
- Ecrire une fonction implique(p,q) en Python qui retourne non p ou q, p et q étant deux booléens.

* Equivalence logique
http://fr.wikipedia.org/wiki/%C3%89quivalence_logique
(p ⇒ q) ET ( q ⇒ p)
- Dresser à l'aide de Python la table de vérité de l'équivalence logique.
- Ecrire une fonction equivalence(p,q) en Python qui retourne (p ⇒ q) ET ( q ⇒ p), p et q étant deux
booléens.

* Le OU exclusif (XOR):
soit p, soit q, mais pas les deux à la fois.
http://fr.wikipedia.org/wiki/Ou_exclusif

 (p OU q) ET ¬(p ET q)

Dresser à l'aide de Python la table de vérité du OU exclusif.
Table de vérité de XOR
p       q       A ⊕ B
0       0       0
0       1       1
1       0       1
1       1       0
Application en cryptographie
```

http://fr.wikipedia.org/wiki/Ou_exclusif#Exemple_d.27utilisation_en_cryptographie

Questions :
- Quel serait le message binaire envoyé du futur dans le film "Le code Andromède" s'il était codé à l'aide de la clé de 7 bits K=1001110 ?
- Ecrire une fonction xor(p,q) en Python qui retourne soit p, soit q mais pas les deux à la fois, p et q étant deux booléens.

Récapitulatif :
http://fr.wikipedia.org/wiki/Fonction_logique

Mini-projet :
A rendre par groupe pour jeudi 27 septembre 2012
--> Ecrire un programme Python qui dresse les tables de vérité de non p, p OU q, p ET q telle qu'elles sont écrites sur cette page.
Annexe : formatage d'une chaîne de caractères : lire "Apprendre à programmer avec Python3" page 158 : Formatage des chaînes de caractères.
http://inforef.be/swi/download/apprendre_python3.pdf

--> Découverte des propriétés de l'Algèbre de Boole
p et q sont deux booléens.
Démontrer à l'aide d'un programme Python que :
```
# Règles d'addition (le OU est noté +)
p + 0 = p
p + 1 = 1
p + p = p
p + NON p = 1
# Règles de multiplication (le ET est noté . )
p . 0 = 0
p . 1 = p
p . p = p
p . NON p = 0
# Commutativité
p . q = q . p
# Associativité
p . ( q + r ) = p . q + p . r

#Les lois de MORGAN
p et q sont deux booléens.
Démontrer à l'aide d'un programme Python que :
 NON (p OU q) = NON p ET NON q
 NON (p ET q) = NON p OU NON q
```

Transformer un entier en binaire

$127 = 63*2 + 1$
$63 = 31*2 + 1$
$31 = 15*2 + 1$
$15 = 7*2 + 1$
$7 = 3*2 + 1$
$3 = 1*2 + 1$
$1 = 0*2 + 1$

restes successifs de la division par 2

$127 = (((((1*2 + 1)*2 + 1)*2 + 1)*2 + 1)*2 + 1)*2 + 1$

$127 = 1*2^6 + 1*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0$

Ainsi 127 s'écrit 1111111 en binaire.

```
m ← 127
c ← ''
Tant que m ≠ 0 Faire:
    c ← c + str(m%2)
    m ← m/2
Fintantque
Afficher c
```

127%2 = 1

'1'

Extrait de la table ASCII (0 - 127)

Décimal	Octal	Hex	Binaire	Caractère	
048	060	30	00110000	0	
049	061	31	00110001	1	
050	062	32	00110010	2	
051	063	33	00110011	3	
052	064	34	00110100	4	
053	065	35	00110101	5	
054	066	36	00110110	6	
055	067	37	00110111	7	
056	070	38	00111000	8	
057	071	39	00111001	9	
058	072	3A	00111010	:	(colon)
059	073	3B	00111011	;	(semi-colon)
060	074	3C	00111100	<	(less than sign)
061	075	3D	00111101	=	(equal sign)
062	076	3E	00111110	>	(greater than sign)
063	077	3F	00111111	?	(question mark)
064	100	40	01000000	@	(AT symbol)
065	101	41	01000001	A	
066	102	42	01000010	B	
067	103	43	01000011	C	
068	104	44	01000100	D	
069	105	45	01000101	E	
070	106	46	01000110	F	
071	107	47	01000111	G	
072	110	48	01001000	H	
073	111	49	01001001	I	
074	112	4A	01001010	J	
075	113	4B	01001011	K	
076	114	4C	01001100	L	
077	115	4D	01001101	M	
078	116	4E	01001110	N	
079	117	4F	01001111	O	
080	120	50	01010000	P	
081	121	51	01010001	Q	
082	122	52	01010010	R	
083	123	53	01010011	S	
084	124	54	01010100	T	
085	125	55	01010101	U	
086	126	56	01010110	V	
087	127	57	01010111	W	
088	130	58	01011000	X	
089	131	59	01011001	Y	
090	132	5A	01011010	Z	

Devoir Informatique et Sciences du Numérique

TS2 - Jeudi 27 septembre 2012 - 2h

Calculatrices autorisées -
Pas d'ordinateur

Première partie : Les bases : système binaire, décimal, hexadécimal

1. Remplir la table de correspondance suivante :

hexadécimal	décimal	binaire
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
A		
B		
C		
D		
E		
F		

2. Donner en binaire les nombres décimaux suivants :
   7 =
   13 =
   256 =
   789 =
   1245 =

3. Convertir en système décimal les entiers suivants donnés en hexadécimal :
   A4F7 =
   BB =
   456 =
   67FA =
   FFFF =

4. a) Quel est le nombre décimal codé par l'octet : 11010110 ?

   b) Convertir 11010110 en hexadécimal :

   c) Ecrire le nombre hexadécimal 7FA6 en binaire :

5. Quel est le plus grand entier naturel que l'on peut coder
   - sur 2 octets :
   - puis sur 4 octets :

6. A l'aide de l'extrait de la table du code ASCII donnée en annexe, décoder le message binaire suivant:
   01001001 01010011 01001110 00110010 00110000 00110001
   00110010 00111010 00110010 00110000 00110001 00110011

7. Ajouter en binaire les nombres suivants : 00111011 et 10001111
   00111011 + 10001111 =

Deuxième partie : Algèbre de Boole

1. Remplir les tables de vérité des opérateurs booléens NON, ET, OU et XOR.
   On rappelle que XOR est le OU exclusif.

p	NON p
Vrai	
Faux	

p	q	p ET q	p OU q	p XOR q
Vrai	Vrai			
Vrai	Faux			
Faux	Vrai			
Faux	Faux			

2. Soit la fonction booléenne suivante : NON(p)·q + p·NON(q)
   (. signifie ET, + signifie OU)

Déterminer la table de vérité de cette fonction :

p	q	NON(p)·q	p·NON(q)	NON(p)·q + p·NON(q)
Vrai	Vrai			
Vrai	Faux			
Faux	Vrai			
Faux	Faux			

Que remarquez-vous ?

Troisième partie : Python

1. 1er exercice : Ecrire les réponses manquantes de Python

```
>>> x=145
>>> y=132
>>> x==y
----
>>> x!=y
----
>>> x>y
----
>>> x>=y
----
>>> x<y
----
>>> x<=y
----
>>> x is y
----
>>> x is not y
----

>>> -5<-6
----
>>> not -5<-6
----

>>> 35<47
----
>>> 87<64
----
>>> (35 < 47) or (87 < 64)
----


>>> 18<19
----
>>> 12<11
----
>>> (18 < 19) and (12 < 11)
----
>>> x=361
>>> (x > 113) and (x < 227)
----
>>> x=200
>>> (x > 113) and (x < 227)
----
```

```
>>>a='dictionnaire1'
>>>b='dictionnaire2'
>>> a==b
----
>>> len(a)
----
>>> a[:12]==b[:12]
----

>>> a='bonjour'
>>> b='toto'
>>> c=a+' ti '+b
>>> c
----
>>> a*3
----


>>> type(False)
----
>>> type(True)
----
>>> bool(' ')
----
>>> bool('')
----
>>> bool(0)
----
>>> bool(156.87)
----


>>> a=['1','toto','350715','227','335','b']
>>> type(a)
----
>>> len(a)
----

>>> for i in range(----,----):
...     print (2**i)
...
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
```

2. 2eme exercice : que font les programmes suivants ?

Programme 1

```
>>> M=1024
>>> x=2
>>> n=0
>>> y=1
>>> while (y<M):
...     y=x*y
...     n=n+1
```

```
...
>>> print n
----
```

programme 2

```
def affiche_PuissanceDe2(a,b):
    for i in range(a,b):
        print(2**i)
affiche_PuissanceDe2(7,13)
```

sortie du programme 2 :

```
----
----
```

3. Ecrire du code Python

Programme 1

Ecrire un programme qui entre 2 entiers naturels x et y. Si x et y sont égaux, ce programme affiche la phrase :
x est égal à y donc la valeur du booléen est : True.
sinon il affiche la phrase :
x est différent de y donc la valeur du booléen est : False.

code Python du programme 1 :

Programme 2

Ecrire un programme qui affiche "Bon anniversaire !" si nous sommes à la date de votre anniversaire, "Bonne journée" sinon.

code Python du programme 2 :

Devoir Informatique et Sciences du Numérique

TS2 - Jeudi 27 septembre 2012 - 2h

Calculatrices autorisées - Pas d'ordinateur

Première partie : Les bases : système binaire, décimal, hexadécimal

1. Remplir la table de correspondance suivante :

base 16 hexadécimal	base 10 décimal	binaire 2 binaire
0	0	0
1	1	1
2	2	10
3		11
4		100
5		101
6		110
7		111
8		1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

2^{10} 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

1024 512 256 128 64 32 16 8 4 2 1

1 1 0 0 0 1 0 1 1 0 1 = 789

$789 = 512 + 256 + 16 + 4 + 1$ (768)

$1245 = 1024 + 128 + 64 + 16 + 8 + 4 + 1$

2. Donner en binaire les nombres décimaux suivants :
   - 7 = 111
   - 13 = 1101
   - 256 = 1 000 000 00
   - 789 = 1100010101
   - 1245 = 10011011101

3. Convertir en système décimal les entiers suivants donnés en hexadécimal :
   - A4F7 = $10\times16^3 + 4\times16^2 + 15\times16^1 + 7\times16^0 = 42231$
   - BB = $11\times16^1 + 11\times16^0 = 187$
   - 456 = 1110
   - 67FA = 26618
   - FFFF = 65535

4. a) Quel est le nombre décimal codé par l'octet : 11010110 ? 214

   b) Convertir 11010110 en hexadécimal : D6 (on écrit 4 par 4 de 4 bits)

   c) Ecrire le nombre hexadécimal 7FA6 en binaire : $7FA6 = 7\times16^3 + 15\times16^2 + 10\times16^1 + 6\times16^0 = 32\,678$

   $32\,678 =$ en binaire 111111101001110

5. Quel est le plus grand entier naturel que l'on peut coder 11111111 sur 8 bits $= 2^8 - 1$
   - sur 2 octets : $2^{16} - 1 = 65535$
   - puis sur 4 octets : 2

6. A l'aide de l'extrait de la table du code ASCII donnée en annexe, décoder le message binaire suivant:

   01001001 01010011 01001110 00110010 00110000 00110001
   00110010 00111010 00110010 00110000 00110001 00110011

   ISN2012:2013

7. Ajouter en binaire les nombres suivants : 00111011 et 10001111

   00111011 + 10001111 = 11001010

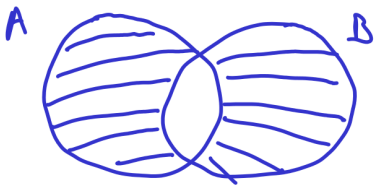
```
  1111 1
  00111011
+ 10001111
----------
  11001010
```

Deuxième partie : Algèbre de Boole

1. Remplir les tables de vérité des opérateurs booléens NON, ET, OU et XOR.
   On rappelle que XOR est le OU exclusif.

p	NON p
Vrai	Faux
Faux	Vrai

p	q	p ET q	p OU q	p XOR q
Vrai	Vrai	Vrai	Vrai	Faux
Vrai	Faux	Faux	Vrai	Vrai
Faux	Vrai	Faux	Vrai	Vrai
Faux	Faux	Faux	Faux	Faux



2. Soit la fonction booléenne suivante : NON(p)·q + p·NON(q)
(. signifie ET, + signifie OU)

Déterminer la table de vérité de cette fonction :

p	q	NON(p)·q (ET)	p·NON(q) (ET)	NON(p)·q + p·NON(q) (OU)
Vrai	Vrai	Faux	Faux	Faux
Vrai	Faux	Faux	Vrai	Vrai
Faux	Vrai	Vrai	Faux	Vrai
Faux	Faux	Faux	Faux	Faux

Que remarquez-vous ? p XOR q = (non p ET q) OU (p ET non q)

Troisième partie : Python

1. 1er exercice : Ecrire les réponses manquantes de Python

```
>>> x=145
>>> y=132
>>> x==y
----False
>>> x!=y
----True
>>> x>y
---- True
>>> x>=y
---- True
>>> x<y
---- False
>>> x<=y
---- False
>>> x is y
---- False
>>> x is not y
---- True

>>> -5<-6
---- False
>>> not -5<-6
---- True

>>> 35<47
---- True
>>> 87<64
---- False
>>> (35 < 47) or (87 < 64)
----  True


>>> 18<19
---- True
>>> 12<11
---- False
>>> (18 < 19) and (12 < 11)
---- False
>>> x=361
>>> (x > 113) and (x < 227)
---- False
>>> x=200
>>> (x > 113) and (x < 227)
---- True
```

```
>>>a='dictionnaire1'
>>>b='dictionnaire2'
>>> a==b
---False
>>> len(a)
----13
>>> a[:12]==b[:12]
----True

>>> a='bonjour'
>>> b='toto'
>>> c=a+' ti '+b
>>> c
----bonjour␣ti␣toto
>>> a*3
----bonjourbonjourbonjour


>>> type(False)
----                    <class 'bool'>
>>> type(True)
----                    <class 'bool'>
>>> bool(' ')
---- True
>>> bool('')
---- False
>>> bool(0)
---- False
>>> bool(156.87)
---- True


>>> a=['1','toto','350715','227','335','b']
>>> type(a)
----                    <class 'list'>
>>> len(a)
----  6

>>> for i in range(---,---):
...     print (2**i)
...
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
```

0, 20 → Python s'arrête à 19 (20-1)

2^0

2^1

2^{19}

2. 2eme exercice : que font les programmes suivants ?

Programme 1

```
>>> M=1024
>>> x=2
>>> n=0
>>> y=1
>>> while (y<M):
...     y=x*y
...     n=n+1
```

Python teste si $1<1024$: vrai

$y = 2 \times y$

n est 1 compteur

Le programme calcule les puissances de 2 jusqu'à $2^{10} = 1024$

```
...
>>> print n
----  n=10
```

programme 2

Le programme calcule les puissances de 2 de 2^7 à 2^{12}

```
def affiche_PuissanceDe2(a,b):
    for i in range(a,b):
        print(2**i)
affiche_PuissanceDe2(7,13)
```

} déf° de la fonction

← programme principal : appel à la fonction.

sortie du programme 2 :

```
----  128
----  256
      512
      1024
      2048
      4096
```

3. Ecrire du code Python

Programme 1

Ecrire un programme qui entre 2 entiers naturels x et y. Si x et y sont égaux, ce programme affiche la phrase :
x est égal à y donc la valeur du booléen est : True.
sinon il affiche la phrase :
x est différent de y donc la valeur du booléen est : False.

code Python du programme 1 :

```
x = input('Entrez 1 entier naturel')
y = input('Entrez 1 entier naturel')
if (x==y):
    print('x est égal à y donc la valeur du booléen est :', bool(x==y))
else:
    print('x est différent de y donc la valeur du booléen est :', bool(x==y))
```

Programme 2

Ecrire un programme qui affiche "Bon anniversaire !" si nous sommes à la date de votre anniversaire, "Bonne journée" sinon.

code Python du programme 2 :

```
date_anniv = input('Entrez votre date d'anniversaire sous la forme jj/mm:')

date_jour = input('Entrez la date d'aujourd'hui sous la forme jj/mm:')

if date_annir == date_jour :
    print('Bon anniversaire')
else:
    print('Bonne journée')
```

CREER SA PREMIERE PAGE HTML

```
<!DOCTYPE HTML>

<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>

<body>
</body>

</html>
```

déclaration du document

entêtes

corps de la page

Travail pour le 8 novembre 2012

Créer une page web à l'aide d'un éditeur de texte. Votre page s'appellera prenom.html .

Cette page contiendra :

- du texte structuré à l'aide des balises <h1>, <h2>, <p>
- du texte mis en forme à l'aide de la balise <font>
- des images
- des liens vers une autre page web
- des liens vers un fichier local de votre machine
- un ou plusieurs tableaux
- une ou plusieurs listes ordonnées et non ordonnées

Validez la page sur http://validator.w3.org/ .

- Exposés
  - 1 Hairata-Thomas-Anthony : Architecture d'un ordinateur
  - 2 Shane-Emilie-Sandra : Formats de fichiers
  - 3 Faouzia-Farda-Jossia : Licence libre/propriétaire
  - 4 Fabien-Clement-Noorman : Commandes Linux
  - 5 Emmanuel-Tommy : Persistance de l'information

Pages Webs
http://lycee-antoine-roussin.ac-reunion.fr/ISN/2012-2013/PageWeb

Exposés
http://lycee-antoine-roussin.ac-reunion.fr/ISN/2012-2013/Exposes/Exposes.html

Travail pour jeudi 24 janvier 2013

- Projets de groupe

- Page Web à contrôler

- Exposés :
  - à préparer,
  - à modifier et corriger

- Travail Algorithmique et python :

1) Algorithme qui donne la résolution
d'une équation du second degré dans le cas général
(y compris quand le discriminant est négatif)
- variables à préciser
- calculs
- tests, boucles

2) Programme python
avec uniquement des fonctions dont les tâches sont spécifiques

a) une fonction trinome qui renvoie l'expression du trinôme :
f(x)=2x**2+3x+1

b) une fonction image qui calcule l'image d'un nombre réel

c) une fonction racines qui calcule les racines réelles ou complexes

d) une fonction signe qui renvoie le signe du trinôme

e) une fonction main qui sera le programme principal.
La fonction main doit mettre en oeuvre toutes les autres.

Aérer le programme avec des commentaires explicatifs.

```
1 DONNÉES
2   a
3   b
4   c
5 VARIABLES
6   delta
7   z1
8   z2
8 DEBUT ALGORITHME
9   ENTRER a, b, c
10  delta PREND LA VALEUR b²-4*a*c  # On calcule delta, le discriminant, en fonction de A,B et C
11  AFFICHER "delta = "
12  AFFICHER delta
13  SI delta<0 ALORS
14     AFFICHER "deux solutions"  # Lorsque delta est negatif,il y a deux solutions complexes
conjuguées, z1 et z2
15     z1 PREND LA VALEUR (- b - i sqrt(-delta))/(2 * a)
16     z2 PREND LA VALEUR (- b + i sqrt(-delta))/(2 * a)
17     FIN SI
18  SI delta=0 ALORS
19     AFFICHER "une solution"   # Lorsque delta est égale à 0, il y a une solution réelle, z
19     z PREND LA VALEUR -b/(2 * a)
20     FIN SI
21  SI delta>0 ALORS
22     AFFICHER "deux solutions"  # Lorsque delta est positif,il y a deux solutions réelles, z1 et z2
22     z1 PREND LA VALEUR (- b - sqrt(delta))/(2 * a)
23     z2 PREND LA VALEUR (- b + sqrt(delta))/(2 * a)
24     FIN SI
25  AFFICHER z1
26  AFFICHER z2
27 FIN ALGORITHME
```

ShaneTrinomeFonction.py

```
# -*- coding: utf-8 -*-

import math

def trinome(a,b,c):
    print("f(x)=",a,"x²","+",b,"x","+",c) #On affiche le polynome de 2nd degré

def image(a,b,c,x):
    print("f(",x,")=",a*x**2+b*x+c) #On calcule l'image de x

def signe(a,b,c,x):
    print ("f(x)=",a,"x²","+",b,"x","+",c)
    if a>0:
        print ("La fonction est du signe de a:",a,"(positive) à l'exeterieur des racines et de signe opposé à l'interieur des racines")
    if a<0:
        print ("La fonction est du signe de a:",a,"(negative) à l'exeterieur des racines et de signe opposé a l'intérieur des racines")

def racines(a,b,c,x):
    d=b*b-4*a*c
    print ("Delta=",d) #On affiche le résultat de delta apres l'avoir calculé
    if d>0:
       print ("Les racines sont ds le plan réel car delta est positif")
       print("x1=",(-b-math.sqrt(d))/(2*a))
       print("x2=",(-b+math.sqrt(d))/(2*a)) #Ce sont les racines de la fonction ds le plan réel
    if d<0:
       print ("Les racines sont ds le plan complexe car delta est negatif")
       print ("x1=",(-b/(2*a)),("+"),math.sqrt(-1*d)/(a*2),"i")
       print ("x2=",(-b/(2*a)),("-"),math.sqrt(-1*d)/(a*2),"i") #Ce sont les racines de la fonction ds le plan complexe

def main(a,b,c,x):
    trinome(a,b,c)
    image(a,b,c,x)
    signe(a,b,c,x)
    racines(a,b,c,x)

main(2,4,5,8) #Cette fonction regroupe l'ensemble de toute les mini-fonction faites auparavant avec les meme coefficients partout
```

```python
import math

def trinome(a,b,c):
    print("f(x)=",a,"x²","+",b,"x","+",c) #On affiche le polynome de 2nd degré

def image(a,b,c,x):
    print("f(",x,")=",a*x**2+b*x+c) #On calcule l'image de x

def signe(a,b,c,x):
    print ("f(x)=",a,"x²","+",b,"x","+",c)
    if a>0:
        print ("La fonction est du signe de a:",a,"(positive) à l'exeterieur des racines et de signe opposé à l'interieur des racines")
    if a<0:
        print ("La fonction est du signe de a:",a,"(negative) à l'exeterieur des racines et de signe opposé a l'intérieur des racines")

def racines(a,b,c,x):
    d=b*b-4*a*c
    print ("Delta=",d) #On affiche le résultat de delta apres l'avoir calculé
    if d>0:
        print ("Les racines sont ds le plan réel car delta est positif")
        print("x1=",(-b-math.sqrt(d))/(2*a))
        print("x2=",(-b+math.sqrt(d))/(2*a)) #Ce sont les racines de la fonction ds le plan réel
    if d<0:
        print ("Les racines sont ds le plan complexe car delta est negatif")
        print ("x1=",(-b/(2*a)),("+"),math.sqrt(-1*d)/(a*2),"i")
        print ("x2=",(-b/(2*a)),("-"),math.sqrt(-1*d)/(a*2),"i") #Ce sont les racines de la fonction ds le plan complexe


def main(a,b,c,x):
    trinome(a,b,c)
    image(a,b,c,x)
    signe(a,b,c,x)
    racines(a,b,c,x)


main(2,4,5,8) #Cette fonction regroupe l'ensemble de toute les mini-fonctions faites auparavant avec les memes coefficients partout
```

PROJET numéro 1 : Réaliser une simulation de feux de forêts en python
Groupes
1. Oummeymane, Sandra, Shane
2. Faouzia, Farda, Jossia

PROJET numéro 2 : Réaliser un jeu de bataille navale en python
Groupes
3. Anthony, Haïrata, Thomas
4. Emilie, Emmanuel, Tommy

PROJET numéro 3 : Mise en oeuvre et programmation du robot Mindstorm
- dans un premier temps à l'aide du logiciel fourni avec le kit robot. Ce logiciel (Windows) sera lancé dans une machine virtuelle sous Linux qui émule Windows XP.
- en deuxième partie, programmation python du robot sous Linux
Groupe
5. Clément, Fabien, Noorman

Vous pouvez commencer à faire des recherches dans Google. Il faut faire preuve d'initiatives.
Veuillez noter toutes les sources des documents et programmes que vous trouverez sur le Net. Soyez donc très méthodiques.
Par exemple, vous pouvez vous envoyer les liens des pages qui vous intéressent sur votre boîte mail et, en parallèle, avoir un petit carnet de bord dans lequel vous noterez à chaque fois la date, et un résumé de vos recherches, avec les liens.

Cela sera essentiel d'avoir cette traçabilité de vos recherches lors de la rédaction du projet final.

Pour démarrer :
projet 1) regardez des vidéos simulant la propagation d'1 feu de forêt
Faire des recherches pour obtenir des modèles les + simples possibles
La question clé est :
Comment se propage 1 feu de forêt ?
une fois un modèle très simple établi, écrire un algorithme le mettant en œuvre.
Implémenter cet algorithme en langage Python.
Finalisez alors votre logiciel de simulation.

projet 2) Trouvez les règles du jeu de la bataille navale.
Télécharger une grille de jeu et commencez par jouer sur le papier entre vous.
Écrire alors un algorithme en 2 temps :
- poser les bateaux sur la grille de manière aléatoire de manière à ce qu'il ne se touchent pas.
On pourra par exemple utiliser un booléen qui sera à 0 si la case est vide, à 1 dès qu'elle est occupée.
- le jeu en lui-même.
Implémenter cet algorithme en langage Python.

Bien sûr, pour vos projets, il existera probablement des programmes déjà écrits en python ou dans 1 autre langage.
Pour les programmes python trouvés, commencez par télécharger les sources et les faire tourner, en installant au préalable les bibliothèques nécessaires. Si vous êtes capables de les comprendre, de décrire leur structure, d'écrire l'algorithme qui a été implémenté et de l'expliquer, pourquoi pas ?
Le programme est utilisable, mais surtout appropriez-vous le, ses variables, ses fonctions. Vous devez pouvoir expliquer chaque variable, le rôle de chaque fonction.

Voir s'il est possible de réaliser un rendu vidéo des sorties de vos programmes, nous pourrons les mettre sur YouTube par exemple.

projet 3) Commencer à réfléchir à ce que vous voulez faire faire au robot. Ecrire les algorithmes correspondants.
Regarder des exemples sur les vidéos youtube.
Commencer à regarder comment se programme le robot. Chercher s'il existe un module de programmation en python et comment on l'installe.

Google Group
"ISN TS2 Lycée Antoine Roussin"

https://groups.google.com/forum/#!forum/isn-ts2-lar

Images des textes en classe

http://nathalierun.net/lycee/piwigo/index.php?/category/50