

Séance 0 : EXPRESSIONS, ÉCRITURE BINAIRE ET LOGIQUE

L1 – Université Côte d’Azur

Exercice 1 — Évaluer des expressions

Quelle est la valeur de chacune des expressions suivantes ?

- 1) `4 // 5 * 3 + 2 ** 3`
- 2) `4 // 5 * (3 + 2) ** 3`
- 3) `2 == 1 + 1`
- 4) `2 == 1 + 1 + 1`
- 5) `(2 == 1 + 1 + 1) and (2 == 1 + 1)`
- 6) `2 == 1 + 1 + 1 or 2 == 1 + 1`
- 7) `1 == 0 // 0`
- 8) `(not (0 == 0)) and 1 == 0 // 0`
- 9) `"2" == '1' + "1"`

Exercice 2 — Écriture binaire

1. En écrivant 14 comme une somme de puissance de 2, le convertir en binaire.
2. Écrire le nombre 11101_2 comme une somme de puissance de 2 ? Combien vaut-il ?
3. Si n est un entier, que représente, au niveau des chiffres, le résultat des opérations $n//10$ et $n\%10$?
4. Si n est un entier, que représente, au niveau des chiffres, le résultat des opérations $n//2$ et $n\%2$?
5. En appliquant l’algorithme vu lors du cours magistral, convertir le nombre 103 en binaire.
6. En appliquant l’algorithme vu lors du cours magistral, quel est l’entier dont l’écriture en binaire est 10101011_2 ?

Exercice 3 — Un peu de logique

1. Faire la table de vérité de la formule « (not a and b) or a »

a	b	not a	not a and b	(not a and b) or a
T	T	F	F	T
T	F	F	F	T
F	T	T	T	T
F	F	T	F	F

2. Pouvez vous écrire la formule précédente sous une forme plus simple ?

 $a \text{ or } b$

3. Démontrer à l'aide d'une table de vérité que « not (a and b) » est égal à « not a or not b »

a	b	a and b	not (a and b)	not a	not b	not a or not b
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

4. Calculer la table de vérité de la formule

 $P = (a \text{ or not } c) \text{ and } (\text{not } a \text{ or } b) \text{ and } (\text{not } b \text{ or } c)$

a	b	c	not a	not b	not c	a or not c	not a or b	not b or c	P
T	T	T	F	F	F	T	T	T	T
T	T	F	F	F	T	T	T	F	F
T	F	T	F	T	F	T	F	T	F
T	F	F	F	T	T	T	F	T	F
F	T	T	T	F	F	F	T	T	F
F	T	F	T	F	T	T	T	F	F
F	F	T	T	T	F	F	T	T	F
F	F	F	T	T	T	T	T	T	T

Exercice 4 — Écrire de petites fonctions

On rappelle que l'on n'a le droit d'utiliser que les outils vu en cours : les conditionnels (**if**), les boucles (**for** ou **while**) et les variables sont strictement interdites !

Pour chacune des fonctions suivantes on indiquera sa signature, c'est-à-dire les types valables pour les paramètres ainsi que le type du résultat.

1. Écrire une fonction
- est_pair**
- (n) qui indique si n est pair.

```
1 # int -> bool
2 def est_pair(n):
3     return n%2 == 0
```

2. Une année est bissextile si elle est multiple de 4 et non multiple de 100 (sauf si multiple de 400 et dans ce cas, elle est bien bissextile). Écrire la fonction
- bissextile**
- (année) qui indique si une année l'est ou non.

```
1 # int -> bool
2 def bissextile(année):
3     return (année%4 == 0 and année%100 != 0) or année%400 == 0
```

3. Écrire une fonction
- année_normale**
- (année) qui indique si une année n'est pas bissextile. On ne pourra pas utiliser la fonction précédente ni utiliser l'opérateur
- not**
- .

L'objectif de la question est de travailler les lois de Morgan

```
1 # int -> bool
2 def année_normale(année):
3     return (année%4 != 0 or année%100 == 0) and année%400 != 0
```

4. Écrire une fonction `est_majeur(j0,m0,a0,j1,m1,a1)` qui pour une personne née le `j0/m0/a0` indique si elle est majeure à la date du `j1/m1/a1`.

```
1 # (int,int,int,int,int,int)-> bool
2 def majeur(j0,m0,a0,j1,m1,a1):
3     return a1>a0+18 or (a1==a0+18 and m1>m0) or (a1==a0+18 and m1==m0 and j1>=j0)
```

5. Écrire (mais sans utiliser `not` ni la fonction `est_majeur`) la fonction `est_mineur(j0,m0,a0,j1,m1,a1)`

```
1 # (int,int,int,int,int,int)-> bool
2 def mineur(j0,m0,a0,j1,m1,a1):
3     return a1<=a0+18 and (a1!=a0+18 or m1<=m0) and (a1 != a0+18 or m1!=m0 or j1<j0)
```

6. Écrire une fonction `bonjour(prénom, nom)` qui à partir d'un prénom et un nom renvoie une phrase de la forme « Bonjour Nicolas Bourbaki ! »

```
1 # (str,str) -> str
2 def bonjour(prénom, nom):
3     return "Bonjour " + prénom + " " + nom + " !"
```

7. Écrire une fonction `répétition(mot, n)` qui répète `n` fois `mot` en séparant par des virgules. Exemple, à partir du mot « hi » et du nombre 5 on obtient le texte « hi, hi, hi, hi, hi ».

```
1 # (str,int) -> str
2 # Ne marche pas pour n <= 0
3 def répétition(mot,n):
4     return (n-1) * (mot + ", ") + mot
```

Exercice 5 — Base 16

La base 16, comme son nom l'indique, utilise 16 chiffres hexa : de 0 à 9 puis de *A* à *F*.

1. Convertir 45 en base 16.

$$45 = 2 \times 16 + 13 = 2D_{16}$$

2. Que vaut le nombre $3A_{16}$?

$$3A_{16} = 3 \times 16 + 10 = 48 + 10 = 58$$

3. On souhaite écrire un nombre entre 0 et 15 en base 2. De combien de bits (chiffre binaire) a-t-on besoin ?

Sur 4 bits on peut représenter les nombres 0 à $1111_2 = 8 + 4 + 2 + 1 = 15$ (16 valeurs différentes)

4. Pour coder un nombre écrit avec deux chiffres hexadécimaux, de combien de bits a-t-on besoin ? Comment appelle-t-on un tel nombre ?

Sur 8 bits on peut représenter 16×16 valeurs, soit 256 (de 0 à 255). On parle d'un octet.

5. Traduire $CAFE_{16}$ en binaire.

On remarque que l'on peut coder chiffre par chiffre.

$$- C = 12 = 1100_2$$

$$- A = 10 = 1010_2$$

$$- F = 15 = 1111_2$$

$$- E = 14 = 1110_2$$

On final on a $CAFE = 1100101011111110$