Devoir surveillé n°4 – NSI
16/11/2023

Exercice 1 (4 points)

1. Déterminer les tables de vérité des deux formules suivantes :
   (a) `a and not(b)`
   (b) `not(not(a) or b)`
2. Ces deux expressions sont-elles équivalentes ?

Exercice 2 (2,5 points)

Écrire un code Python qui définit par compréhension la liste `l` contenant les cubes (exposant 3) des entiers de 5 à 125.
Autrement dit, la liste `l` doit contenir les nombres 5^3, 6^3, ..., 125^3.

Exercice 3 (3,5 points)

1. Réécrire et compléter la fonction `contient` ci-dessous, qui prend en argument une liste `l` et une valeur `v`, et qui retourne `True` si `l` contient la valeur `v`, et qui retourne `False` sinon.

```
def contient(l,v):
  for x in l:
    if ... == v:
      ........
  ........
```

2. Réécrire et compléter le code suivant de la fonction `effaceDoublons` qui prend en argument une variable `l` de type `list` et qui retourne une nouvelle liste contenant les éléments de `l` mais sans doublon.

```
def effaceDoublons(l):
  nouvelle_liste = []
  for i in range(len(l)):
    if not(contient(...,...)):
      ........
  return ........
```

Par exemple, `effaceDoublons([2,4,5,2,1,4,7])` retourne la liste `[2,4,5,1,7]`.

Devoir surveillé n°4 – NSI
16/11/2023

Exercice 1 (4 points)

1. Déterminer les tables de vérité des deux formules suivantes :
   (a) `not(a) or b`
   (b) `not(a and not(b))`
2. Ces deux expressions sont-elles équivalentes ?

Exercice 2 (2,5 points)

Écrire un code Python qui définit par compréhension la liste `l` contenant les puissances de 2, les exposants allant de 0 à 20.
Autrement dit, la liste `l` doit contenir les nombres 2^0, 2^1, ..., 2^{20}.

Exercice 3 (3,5 points)

1. Réécrire et compléter la fonction `contient` ci-dessous, qui prend en argument une liste `l` et une valeur `v`, et qui retourne `True` si `l` contient la valeur `v`, et qui retourne `False` sinon.

```
def contient(l,v):
  for i in range(len(l)):
    if ... == v:
      ........
  ........
```

2. Réécrire et compléter le code suivant de la fonction `effaceDoublons` qui prend en argument une variable `l` de type `list` et qui retourne une nouvelle liste contenant les éléments de `l` mais sans doublon.

```
def effaceDoublons(l):
  nouvelle_liste = []
  for x in l:
    if not(contient(...,...)):
      ........
  return ........
```

Par exemple, `effaceDoublons([2,4,5,2,1,4,7])` retourne la liste `[2,4,5,1,7]`.