CoursRepr?sentationNombres2019-2020

December 1, 2019

0.0.1 Représentation des nombres

0.1 Encodage des entiers naturels

0.1.1 Exercice 1

```
In [3]: def chiffres2nombre(t):
            """Retourne l'écriture en base 10 d'un entier à partir de
            la liste de ses chiffres"""
            n = 0
            expomax = len(t) - 1
            for k in range(expomax + 1):
                n = n + 10 ** (expomax - k) * t[k]
            return n

In [4]: chiffres2nombre([7,3,4])

Out[4]: 734
```

```
In [3]: def horner(liste):
            """Retourne un entier à partir de la liste de ses chiffres
            ordonnée par poids décroissant"""
            nombre = 0
            for chiffre in liste:
                nombre = nombre * 10
                nombre = nombre + chiffre
            return nombre

In [2]: horner([7,3,4])

Out[2]: 734
```

0.2 Exercice 2

```
In [1]: def listeChiffre(n):
            """Retourne la liste de chiffres en base 10 d'un entier n"""
            leschiffres = []
            while n >= 10:
                chiffre = n % 10
```

```
            leschiffres.append(chiffre)
            n = n // 10
        leschiffres.append(n)
        leschiffres.reverse()
        return leschiffres
```

```
In [8]: listeChiffre(734)
```

```
Out[8]: [7, 3, 4]
```

0.3 Exercice 3

```
In [6]: def bits2nombre(t):
            """Retourne l'entier en base dix représenté
            par une liste de bits t avec bits de poids fort
            à gauche"""
            n = 0
            for bit in t:
                n = n * 2 + bit
            return n
```

0.4 Exercice 4

```
In [25]: def codageBinaireGlouton(n):
             binaire = []
             puissance2 = 1
             while puissance2 <= n:
                 puissance2 = puissance2 * 2
             puissance2 = puissance2 // 2
             #puissance2 contient alors la plus grande puissance de 2 <= n
             while puissance2 >= 2:
                 if puissance2 <= n:
                     binaire.append(1)
                     n = n - puissance2
                 else:
                     binaire.append(0)
                 puissance2 = puissance2 // 2
             binaire.append(n)
             return binaire
```

```
In [26]: [codageBinaireGlouton(n) for n in range(0, 17)]
```

```
Out[26]: [[0],
          [1],
          [1, 0],
          [1, 1],
          [1, 0, 0],
          [1, 0, 1],
          [1, 1, 0],
```

```
           [1, 1, 1],
           [1, 0, 0, 0],
           [1, 0, 0, 1],
           [1, 0, 1, 0],
           [1, 0, 1, 1],
           [1, 1, 0, 0],
           [1, 1, 0, 1],
           [1, 1, 1, 0],
           [1, 1, 1, 1],
           [1, 0, 0, 0, 0]]
```

```
In [27]: def codageBinaire2(n):
             binaire = []
             while n >= 2:
                 binaire.append(n % 2)
                 n = n // 2
             binaire.append(n)
             binaire.reverse()
             return binaire
```

```
In [29]: [codageBinaire2(n) for n in range(0, 17)]
```

```
Out[29]: [[0],
          [1],
          [1, 0],
          [1, 1],
          [1, 0, 0],
          [1, 0, 1],
          [1, 1, 0],
          [1, 1, 1],
          [1, 0, 0, 0],
          [1, 0, 0, 1],
          [1, 0, 1, 0],
          [1, 0, 1, 1],
          [1, 1, 0, 0],
          [1, 1, 0, 1],
          [1, 1, 1, 0],
          [1, 1, 1, 1],
          [1, 0, 0, 0, 0]]
```

0.5 Exercice 5

```
In [8]: def additionBinaire8bits(t1, t2):
            t3 = [0]  * 8
            retenue = 0
            for k in range(7, -1 , -1):
                s = t1[k] + t2[k] + retenue
                t3[k] =  s % 2
```

```
            retenue = s // 2
        return t3
```

```
In [10]: additionBinaire8bits([1,0,1,0,1,1,1,0],[0,0,0,0,1,1,1,1])
```

```
Out[10]: [1, 0, 1, 1, 1, 1, 0, 1]
```

0.6 Exemple de dépassement de capacité pour des entiers non signés sur 8 bits

```
In [31]: import numpy as np
```

```
In [33]: np.uint8(255) + np.uint8(1)
```

```
/home/fjunier/.local/lib/python3.6/site-packages/ipykernel_launcher.py:1: RuntimeWarning: overi
  """Entry point for launching an IPython kernel.


Out[33]: 0
```

0.7 Représentation des réels

0.8 Exercice 9

```
In [3]: def decomposition_fraction_egyptienneV1(x):
            """Retourne une liste de dénominateurs entiers p1,, p2, ..., pn
            tels que x = 1/p1 + 1/ p2 + .... + 1 / pn"""
            decomp = []
            while x != 0:
                n = 1
                while 1/n > x:
                    n += 1
                decomp.append(n)
                x = x - 1 / n
                print(x, decomp)  #pour le débogage
            return decomp
```

```
In [4]: import math

        def decomposition_fraction_egyptienneV2(x):
            """Retourne une liste de dénominateurs entiers p1,, p2, ..., pn
            tels que x = 1/p1 + 1/ p2 + .... + 1 / pn"""
            decomp = []
            while x != 0:
                n = math.ceil(1/x)
                decomp.append(n)
                x = x - 1 / n
                print(x, decomp)  #pour le débogage
            return decomp
```

```
In [9]: decomposition_fraction_egyptienneV1(3/4)
```

```
0.25 [2]
0.0 [2, 4]
```

Out[9]: [2, 4]