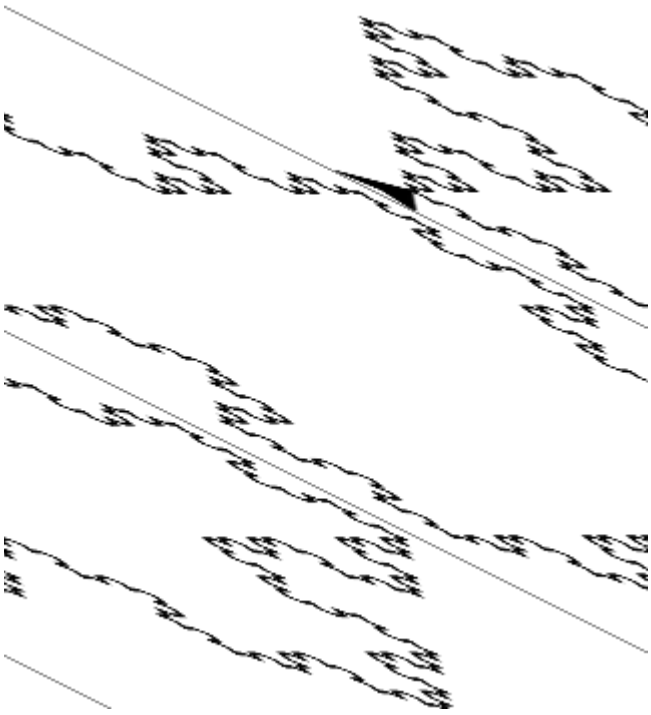
- TP 5 -
Programmation récursive

par Édouard Lumet



Sommaire

1. Introduction – problème complexe....3
2. Décompositions en problèmes simples....4
    2.1. Étape 1 : écriture des chiffres....4
    2.2. Étape 2 : nombres particuliers de 10 à 19....4
    2.3. Étape 3 : écriture des dizaines....4
    2.4. Étape 4 : centaines et multiples de mille....4
3. Programme complet....5

1. Introduction – problème complexe

L'objectif de ce programme est de convertir un nombre entier quelconque en son libellé en français. Par exemple, ce programme doit retourner « deux mille quatre cent cinquante six » si on l'appel avec 2456. La forme sera la suivante :

Entrez un nombre : **2456**
Son écriture est : **deux mille quatre cent cinquante six**

Les objectifs d'apprentissage quant à eux sont :

- savoir programmer une exception (*try…*),
- programmer des algorithmes récursifs (fonction faisant appel à elle-même dans sa définition),
- être capable de décomposer un problème complexe en plusieurs problèmes simples,
- savoir réaliser une itération à l'aide de la récursivité et non à l'aide d'une boucle.

2. Décomposition en problèmes simples

2.1. Étape 1 : écriture des chiffres

Dans un premier temps, on écrit une fonction *nom_chiffre* qui transforme les chiffres en leur nom français. On utilise pour cela un dictionnaire associant chaque chiffre à son libellé. Enfin, on programme une exception pour tester si le paramètre n'est pas un chiffre.

```
#/usr/bin/python2.7
# -*- coding: utf-8 -*-

def nom_chiffre(c):
    try:
        if c < 0 or c > 9:
            raise ValueError
    except ValueError:
        print "Veuillez entrer un CHIFFRE"
        quit()

    libelle={1: "un", 2: "deux", 3: "trois", 4: "quatre", 5: "cinq", 6: "six", 7: "sept", 8: "huit", 9: "neuf"}
    if c in range(1, 10):
        libelle_chiffre=libelle.get(c)
    return libelle_chiffre

# Programme principal

print nom_chiffre(5)
```

2.2. Étape 2 : nombres particuliers de 10 à 19

Ensuite, on fait de même concernant les nombres entre 10 et 19 dans une fonction *nom_deDixAVingt*. On programme également une exception pour tester le paramètre entré lors de l'appel de la fonction.

```
#/usr/bin/python2.7
# -*- coding: utf-8 -*-

def nom_deDixAVingt(n):
   try:
      if n < 10 or n > 19:
         raise ValueError
   except ValueError:
      print "Veuillez entrer un nombre entre 10 et 19 inclus"
      quit()

   libelle={10: "dix", 11: "onze", 12: "douze", 13:"treize", 14:"quatorze", 15: "quinze", 16: "seize", 17: "dix-sept",
18: "dix-huit", 19: "dix-neuf"}
   if n in range(10, 20):
      libelle_nombreDixVingt=libelle.get(n)
   return libelle_nombreDixVingt

# Programme principal

print nom_deDixAVingt(18)
```

2.3. Étape 3 : écriture des dizaines

Dans cette troisième étape, on programme une fonction *nom_dizaine* qui transforme les dizaines de 20 à 90. Comme pour les étapes précédentes, on utilise pour cela un dictionnaire et on programme une exception pour tester le nombre entré.

```
#/usr/bin/python2.7
# -*- coding: utf-8 -*-

def nom_dizaine(n):
   try:
      if not n in xrange(20, 100, 10):
         raise ValueError
   except ValueError:
      print "Veuillez entrer un nombre multiple de 10 entre 20 et 90 inclus"
      quit()

   libelle={20: "vingt", 30: "trente", 40: "quarante", 50: "cinquante", 60: "soixante", 70: "soixante-dix", 80: "quatre-vingt", 90: "quatre-vingt-dix"}
   if n in xrange(20, 100, 10):
      libelle_dixaine=libelle.get(n)
   return libelle_dixaine

# Programme principal

print nom_dizaine(30)
```

2.4. Étape 4 : centaines et multiples de mille

```
liste=[]
def nom_un_entier(n):
  try:
    n1=n%1
    if n1 != 0 or n < 0 or n >= 1000000000:
      raise ValueError
  except ValueError:
    print "Veuillez entre un nombre ENTIER entre 0 et 9 999 999 inclus"
    quit()
  if n >= 100000000 and n < 1000000000:
    n1=n%1000000
    n=n/1000000
    liste.append(nom_chiffre(n/100))
    liste.append("cent")
    liste.append(nom_dizaine((n%100)/10*10))
    liste.append(nom_chiffre(n%10))
    liste.append("million")
    nom_un_entier(n1)

  if n >= 1000 and n < 1000000:
    n1=n%1000
    n=n/1000
    liste.append(nom_chiffre(n/100))
    liste.append("cent")
    liste.append(nom_dizaine((n%100)/10*10))
    liste.append(nom_chiffre(n%10))
    liste.append("mille")
    nom_un_entier(n1)

  if n >= 100 and n < 1000:
    liste.append(nom_chiffre(n/100))
    liste.append("cent")
    nom_un_entier(n%100)

  if n >= 20 and n < 100:
    liste.append(nom_dizaine(n/10*10))
    nom_un_entier(n%10)

  if n >= 10 and n < 20:
    liste.append(nom_deDixAVingt(n))
    for e in liste:
      print e,

  if n >= 1 and n < 10:
    liste.append(nom_chiffre(n))
    for e in liste:
      print e,

  if n == 0:
    for e in liste:
      print e,
```

Enfin, on finit par la transformation des centaines et des multiples de mille puis on définit le comportement selon le paramètre de la fonction *nom_un_entier*.

C'est cette fonction que l'on appelle pour traduire n'importe quel nombre en appelant la fonction elle-même puis les fonctions précédentes.

Cette fonction relève donc de la programmation récursive.

NB: A partir de cette étape, trois comportements apparaissent.

- *Lors de l'appel avec le nombre **560874** par exemple, le résultat contient trois fois le libellé.*
- *Lors de l'appel avec le nombre **348** par exemple, le résultat est correct. De même avec les nombres ne faisant appel qu'aux fonctions des étapes 1 à 3 uniquement.*
- *Lors de l'appel avec le nombre **1498** par exemple, l'erreur de l'exception liée aux dizaines apparaît.*

Je n'ai pas réussi à débugger le problème… :-(

3. Programme complet

```
#/usr/bin/python2.7
# -*- coding: utf-8 -*-

def nom_chiffre(c):
  try:
    if c < 0 or c > 9:
      raise ValueError
  except ValueError:
    print "Veuillez entrer un CHIFFRE"
    quit()

  libelle={0: "zéro", 1: "un", 2: "deux", 3: "trois", 4: "quatre", 5: "cinq", 6: "six", 7: "sept", 8: "huit", 9: "neuf"}
  if c in range(0, 10):
    libelle_chiffre=libelle.get(c)
  return libelle_chiffre

def nom_deDixAVingt(n):
  try:
    if n < 10 or n > 19:
      raise ValueError
  except ValueError:
    print "Veuillez entrer un nombre entre 10 et 19 inclus"
    quit()

  libelle={10: "dix", 11: "onze", 12: "douze", 13:"treize", 14:"quatorze", 15: "quinze", 16: "seize", 17: "dix-sept", 18: "dix-
huit", 19: "dix-neuf"}
  if n in range(10, 20):
    libelle_nombreDixVingt=libelle.get(n)
  return libelle_nombreDixVingt

def nom_dizaine(n):
  try:
    if not n in xrange(20, 100, 10):
      raise ValueError
  except ValueError:
    print "Veuillez entrer un nombre multiple de 10 entre 20 et 90 inclus"
    quit()

  libelle={20: "vingt", 30: "trente", 40: "quarante", 50: "cinquante", 60: "soixante", 70: "soixante-dix", 80: "quatre-
vingt", 90: "quatre-vingt-dix"}
  if n in xrange(20, 100, 10):
    libelle_dixaine=libelle.get(n)
  return libelle_dixaine

liste=[]
def nom_un_entier(n):
  try:
    n1=n%1
```

```
    if n1 != 0 or n < 0 or n >= 1000000000:
      raise ValueError
  except ValueError:
    print "Veuillez entre un nombre ENTIER entre 0 et 9 999 999 inclus"
    quit()
  if n >= 100000000 and n < 1000000000:
    n1=n%1000000
    n=n/1000000
    liste.append(nom_chiffre(n/100))
    liste.append("cent")
    liste.append(nom_dizaine((n%100)/10*10))
    liste.append(nom_chiffre(n%10))
    liste.append("million")
    nom_un_entier(n1)

  if n >= 1000 and n < 1000000:
    n1=n%1000
    n=n/1000
    liste.append(nom_chiffre(n/100))
    liste.append("cent")
    liste.append(nom_dizaine((n%100)/10*10))
    liste.append(nom_chiffre(n%10))
    liste.append("mille")
    nom_un_entier(n1)

  if n >= 100 and n < 1000:
    liste.append(nom_chiffre(n/100))
    liste.append("cent")
    nom_un_entier(n%100)

  if n >= 20 and n < 100:
    liste.append(nom_dizaine(n/10*10))
    nom_un_entier(n%10)

  if n >= 10 and n < 20:
    liste.append(nom_deDixAVingt(n))
    for e in liste:
      print e,

  if n >= 1 and n < 10:
    liste.append(nom_chiffre(n))
    for e in liste:
      print e,

  if n == 0:
    for e in liste:
      print e,

# Programme principal

nom_un_entier(356)
```

```
>>> trois cent cinquante six
```