

- Nombre d'occurrence d'un terme dans fichier text avec for et while

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

def countTerm(listEle:List[String], term:String) =
{
  var nbTerm = 0
  var j = 0
  while(j < listEle.length)
  {
    if(term == listEle(j))
      nbTerm+=1
    j += 1
  }
  nbTerm
}

import scala.io.Source

val lines = Source.fromFile("/home/momo/Desktop/Formation_Big_Data/file.txt").getLines.toList

countTerm(lines, "fonction")

// Exiting paste mode, now interpreting.

countTerm: (listEle: List[String], term: String)Int
import scala.io.Source
lines: List[String] = List(Test, fichier, fonction, Scala, Loop, fonction, Collection, fonction)
res14: Int = 3

scala> 
```

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

def countTerm(listEle:List[String], term:String) =
{
  var nbTerm = 0
  for(name <- listEle)
  {
    if(term == name)
      nbTerm+=1
  }
  nbTerm
}

import scala.io.Source

val lines = Source.fromFile("/home/momo/Desktop/Formation_Big_Data/file.txt").getLines.toList

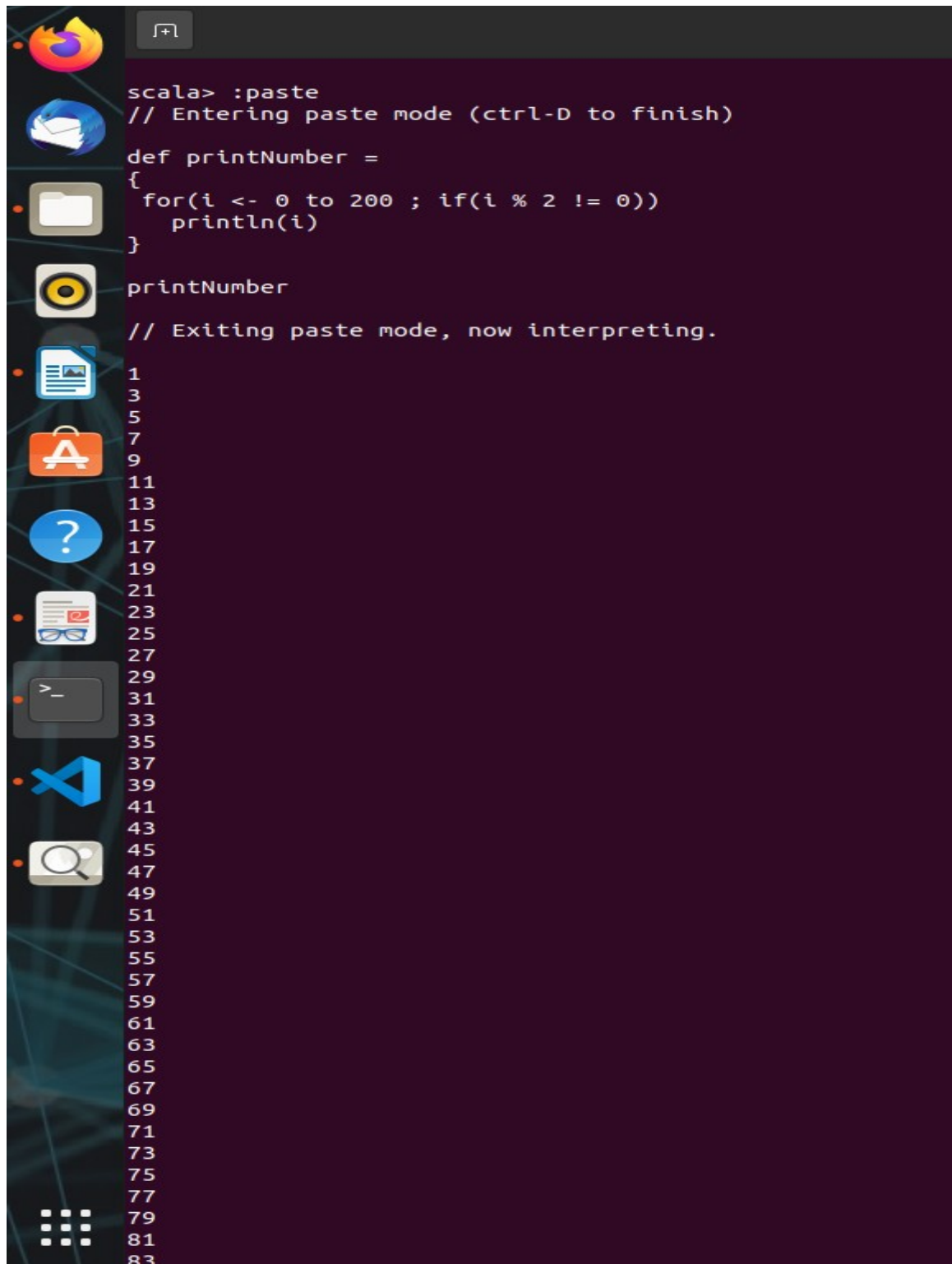
countTerm(lines, "fonction")

// Exiting paste mode, now interpreting.

countTerm: (listEle: List[String], term: String)Int
import scala.io.Source
lines: List[String] = List(Test, fichier, fonction, Scala, Loop, fonction, Collection, fonction)
res10: Int = 3

scala> 
```

- Affichage des nombre impair entre 0 et 200 en utilisant la boucle for

A terminal window with a dark purple background and white text. On the left side, there is a vertical dock with various application icons: Firefox, a mail icon, a folder icon, a CD icon, a document icon, an App Store icon, a question mark icon, a document with glasses icon, a terminal icon, a blue 'X' icon, and a magnifying glass icon. The terminal text shows a Scala REPL session where a function 'printNumber' is defined and then called. The function prints odd numbers from 1 to 83. The text is as follows:

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

def printNumber =
{
  for(i <- 0 to 200 ; if(i % 2 != 0))
    println(i)
}

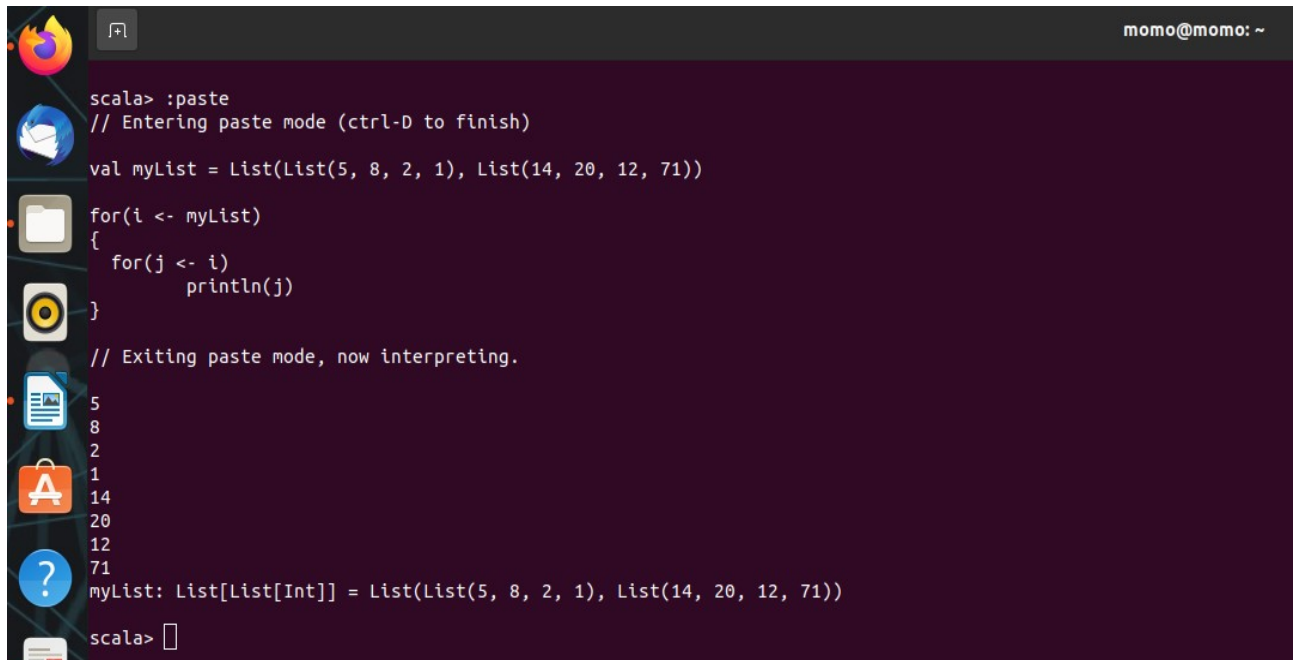
printNumber

// Exiting paste mode, now interpreting.

1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
31
33
35
37
39
41
43
45
47
49
51
53
55
57
59
61
63
65
67
69
71
73
75
77
79
81
83
```

- Lorsqu'on affecte une boucle for ou while à une variable cette dernière sera vide(de type Unit) car une boucle permet de faire une itération et donc ne retourne rien.
- Dans l'exemple ci-dessous, nous avons utilisé deux variable dans une boucle for à l'aide de la fonction zip() utilisée pour fusionner une collection avec la collection actuelle et le résultat est une collection de paires d'éléments de tuple des deux collections.

- Nous avons défini une liste de listes et nous avons une première boucle **for** pour parcourir les listes de la grande liste ensuite une deuxième boucle **for**(imbriquée dans la première boucle) pour parcourir les éléments de chaque sous-liste.



```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val myList = List(List(5, 8, 2, 1), List(14, 20, 12, 71))

for(i <- myList)
{
  for(j <- i)
    println(j)
}

// Exiting paste mode, now interpreting.

5
8
2
1
14
20
12
71
myList: List[List[Int]] = List(List(5, 8, 2, 1), List(14, 20, 12, 71))

scala>
```

- L'algorithme Merge Sort est un algorithme de tri basé sur le paradigme Diviser et Conquérir. Dans cet algorithme, le tableau est d'abord divisé en deux moitiés égales, puis elles sont combinées de manière triée. Il s'agit d'un algorithme récursif qui divise continuellement le tableau en deux jusqu'à ce qu'il ne puisse plus être divisé.