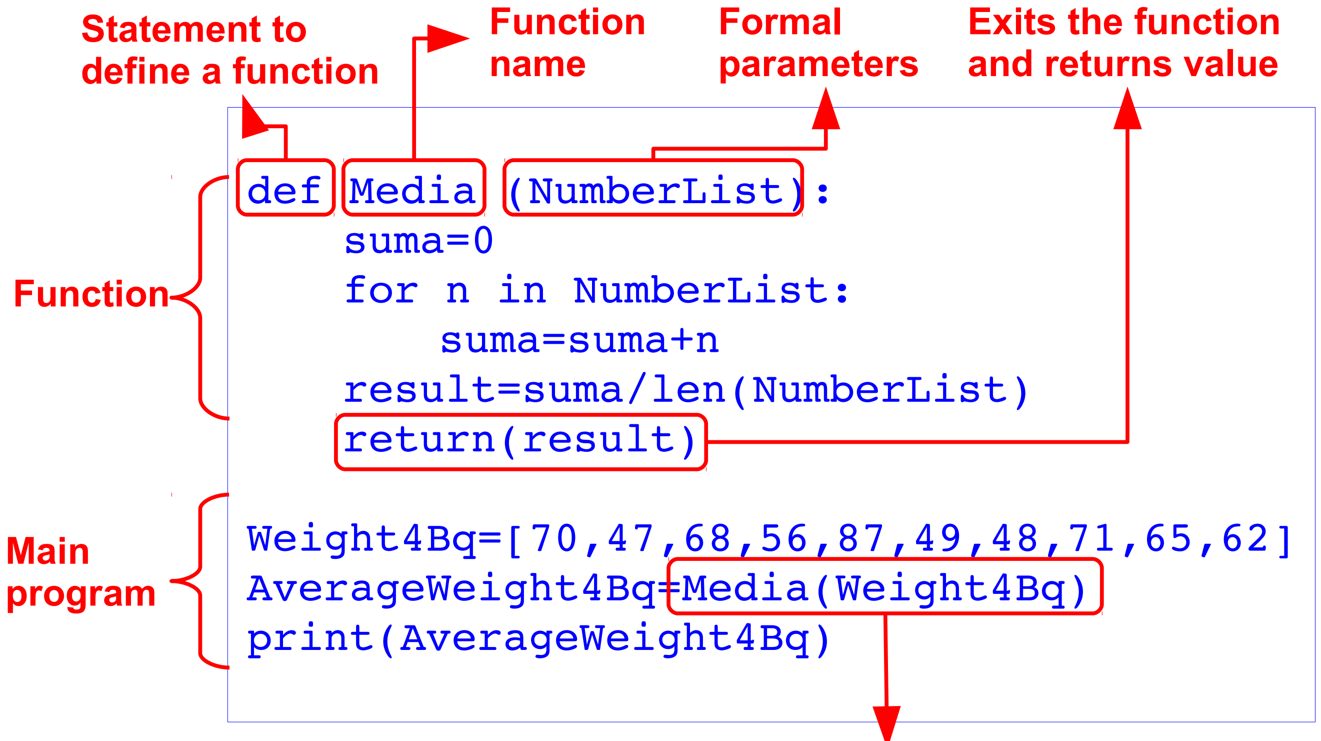
User-defined functions

We can think of a function as a subprogram (a small program inside of a program).
The idea is to create a block of code and give it a name so we can call it from any proint within a program

User defined functions:
1) To avoid code duplications.
2) To make your code modular (divide it into individual tasks so it is tidy and recyclable)

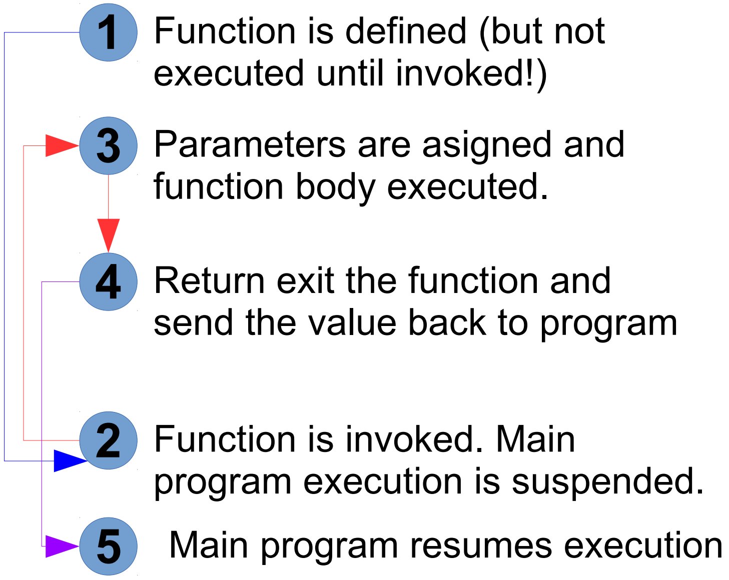

Statement to define a function
Function name
Formal parameters
Exits the function and returns value
Function
Main program

```
def Media (NumberList):
    suma=0
    for n in NumberList:
        suma=suma+n
    result=suma/len(NumberList)
    return(result)

Weight4Bq=[70,47,68,56,87,49,48,71,65,62]
AverageWeight4Bq=Media(Weight4Bq)
print(AverageWeight4Bq)
```

Functions must be defined before they can be invoked

Invoking or calling a function

Order of code execution:



1 Function is defined (but not executed until invoked!)

3 Parameters are asigned and function body executed.

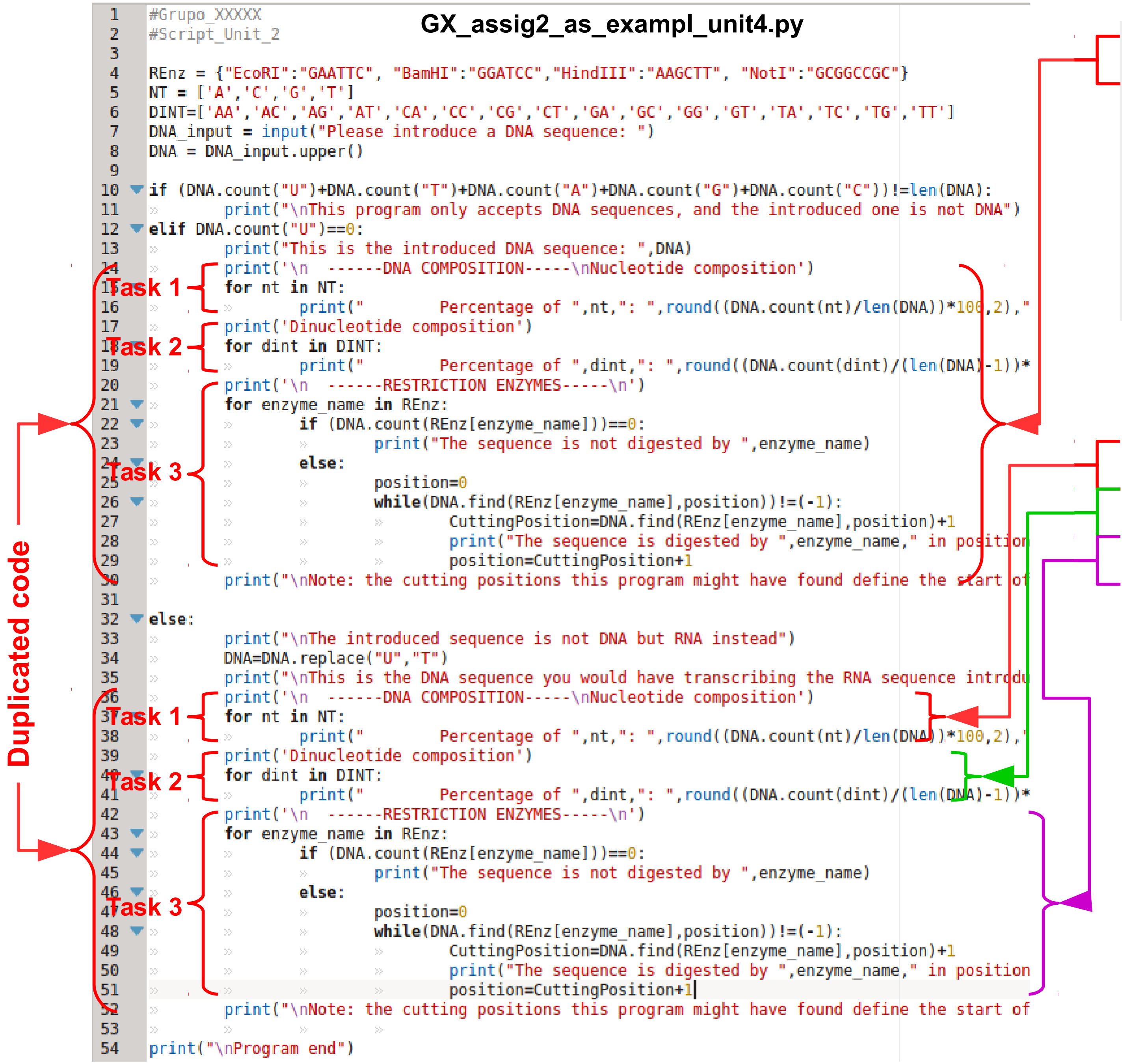
4 Return exit the function and send the value back to program

2 Function is invoked. Main program execution is suspended.

5 Main program resumes execution

Unit4_Functions_example1.py

Example: an script from the Unit2 assignment



GX_assig2_as_exampl_unit4.py

```
#Grupo_XXXXX
#Script_Unit_2

REnz = {"EcoRI":"GAATTC", "BamHI":"GGATCC","HindIII":"AAGCTT", "NotI":"GCGGCCGC"}
NT = ['A','C','G','T']
DINT=['AA','AC','AG','AT','CA','CC','CG','CT','GA','GC','GG','GT','TA','TC','TG','TT']
DNA_input = input("Please introduce a DNA sequence: ")
DNA = DNA_input.upper()

if (DNA.count("U")+DNA.count("T")+DNA.count("A")+DNA.count("G")+DNA.count("C"))!=len(DNA):
        print("\nThis program only accepts DNA sequences, and the introduced one is not DNA")
elif DNA.count("U")==0:
        print("This is the introduced DNA sequence: ",DNA)
        print('\n  ------DNA COMPOSITION-----\nNucleotide composition')
        for nt in NT:
                print("        Percentage of ",nt,": ",round((DNA.count(nt)/len(DNA))*100,2),"
        print('Dinucleotide composition')
        for dint in DINT:
                print("        Percentage of ",dint,": ",round((DNA.count(dint)/(len(DNA)-1))*
        print('\n  ------RESTRICTION ENZYMES-----\n')
        for enzyme_name in REnz:
                if (DNA.count(REnz[enzyme_name]))==0:
                        print("The sequence is not digested by ",enzyme_name)
                else:
                        position=0
                        while(DNA.find(REnz[enzyme_name],position))!=(-1):
                                CuttingPosition=DNA.find(REnz[enzyme_name],position)+1
                                print("The sequence is digested by ",enzyme_name," in position
                                position=CuttingPosition+1
        print("\nNote: the cutting positions this program might have found define the start of

else:
        print("\nThe introduced sequence is not DNA but RNA instead")
        DNA=DNA.replace("U","T")
        print("\nThis is the DNA sequence you would have transcribing the RNA sequence introdu
        print('\n  ------DNA COMPOSITION-----\nNucleotide composition')
        for nt in NT:
                print("        Percentage of ",nt,": ",round((DNA.count(nt)/len(DNA))*100,2),"
        print('Dinucleotide composition')
        for dint in DINT:
                print("        Percentage of ",dint,": ",round((DNA.count(dint)/(len(DNA)-1))*
        print('\n  ------RESTRICTION ENZYMES-----\n')
        for enzyme_name in REnz:
                if (DNA.count(REnz[enzyme_name]))==0:
                        print("The sequence is not digested by ",enzyme_name)
                else:
                        position=0
                        while(DNA.find(REnz[enzyme_name],position))!=(-1):
                                CuttingPosition=DNA.find(REnz[enzyme_name],position)+1
                                print("The sequence is digested by ",enzyme_name," in position
                                position=CuttingPosition+1
        print("\nNote: the cutting positions this program might have found define the start of

print("\nProgram end")
```

GX_assig2_as_exampl_unit4_mod.py

```
Function AnalizeSeq(Seq,Nt,Dn,Re):
        code_block
### main body of the script:
if not DNA:
        exit
elif DNA:
        AnalizeSeq(DNA,NT,DINT,REnz)
else:
        DNA=DNA.replace(“U”,”T”)
        AnalizeSeq(DNA,NT,DINT,REnz)
```

GX_assig2_as_exampl_unit4_mod_v2.py
GX_assig2_as_exampl_unit4_mod_v3.py

```
Function NucletComp(Seq,Nt):
        code_block2
Function DinucletComp(Seq,Dn):
        code_block3
Function RestSites(Seq,Re):
        code_block4

### main body of the script:
if not DNA:
        exit
elif DNA:
        NucletComp(DNA,NT)
        DinucletComp(DNA,DINT)
        RestSites(DNA,REnz)
else:
        DNA=DNA.replace(“U”,”T”)
        NucletComp(DNA,NT)
        DinucletComp(DNA,DINT)
        RestSites(DNA,REnz)
```

Parameter passing: by position or by key-word

Type some spaces Between "..." and commands (INDENTATION)

```
>>>Division(10,5)
>>> def Division (Dividendo, Divisor):
...     return(Dividendo/Divisor)
...
>>>
>>>Division(10,5)
>>>Division(5,10)
>>>Division(Dividendo=10,Divisor=5)
>>>Division(Divisor=5,Dividendo=10)
```

Press enter ONLY (no spaces) to indicate end of code block.

We have to define the function before invoking it.

```
>>>Division(10,5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Division' is not defined
>>> def Division (Dividendo, Divisor):
...     return(result)
...
>>>Division(10,5)
2.0
>>>Division(5,10)
0.5
>>>Division(Dividendo=10,Divisor=5)
2.0
>>>Division(Divisor=5,Dividendo=10)
2.0
```

Arguments by position

Position determines which one is assigned to Divisor and Dividendo

Arguments by keyword

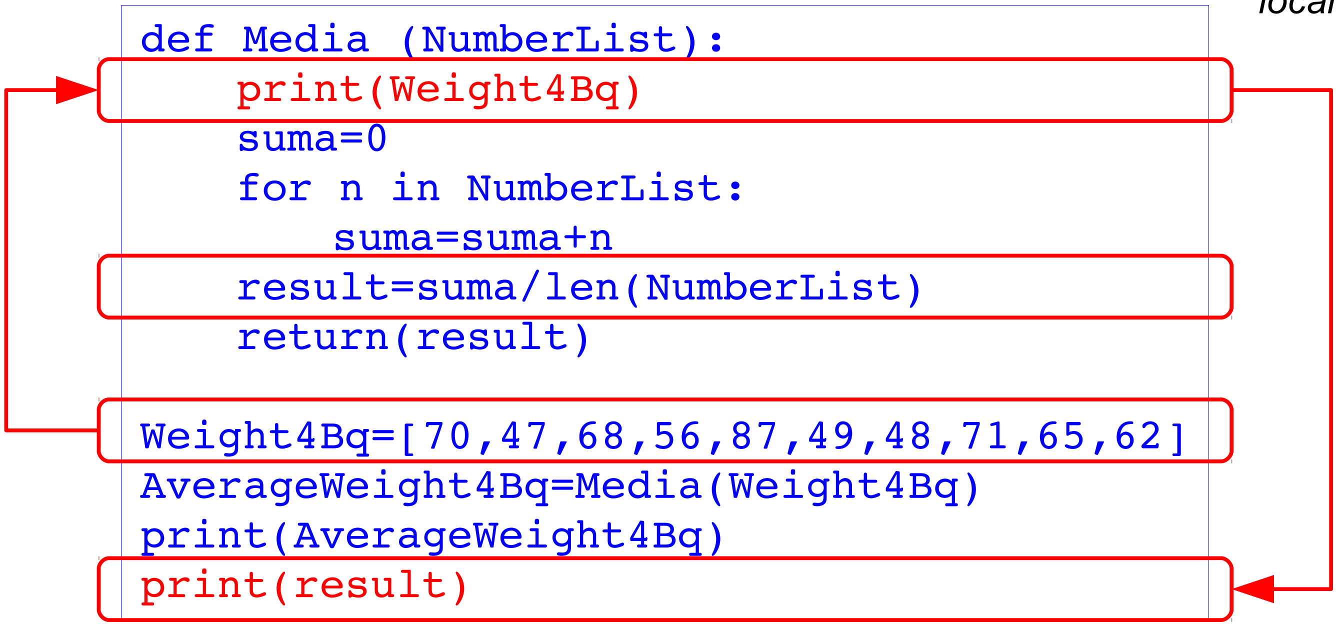
Position doesn't determines which one is assigned to Divisor and Dividendo

Global and local variables

Unit4_Functions_example1.py

```
def Media (NumberList):
    suma=0
    for n in NumberList:
        suma=suma+n
    result=suma/len(NumberList)
    return(result)

Weight4Bq=[70,47,68,56,87,49,48,71,65,62]
AverageWeight4Bq=Media(Weight4Bq)
print(AverageWeight4Bq)
```

Try:



```
def Media (NumberList):
    print(Weight4Bq)
    suma=0
    for n in NumberList:
        suma=suma+n
    result=suma/len(NumberList)
    return(result)

Weight4Bq=[70,47,68,56,87,49,48,71,65,62]
AverageWeight4Bq=Media(Weight4Bq)
print(AverageWeight4Bq)
print(result)
```

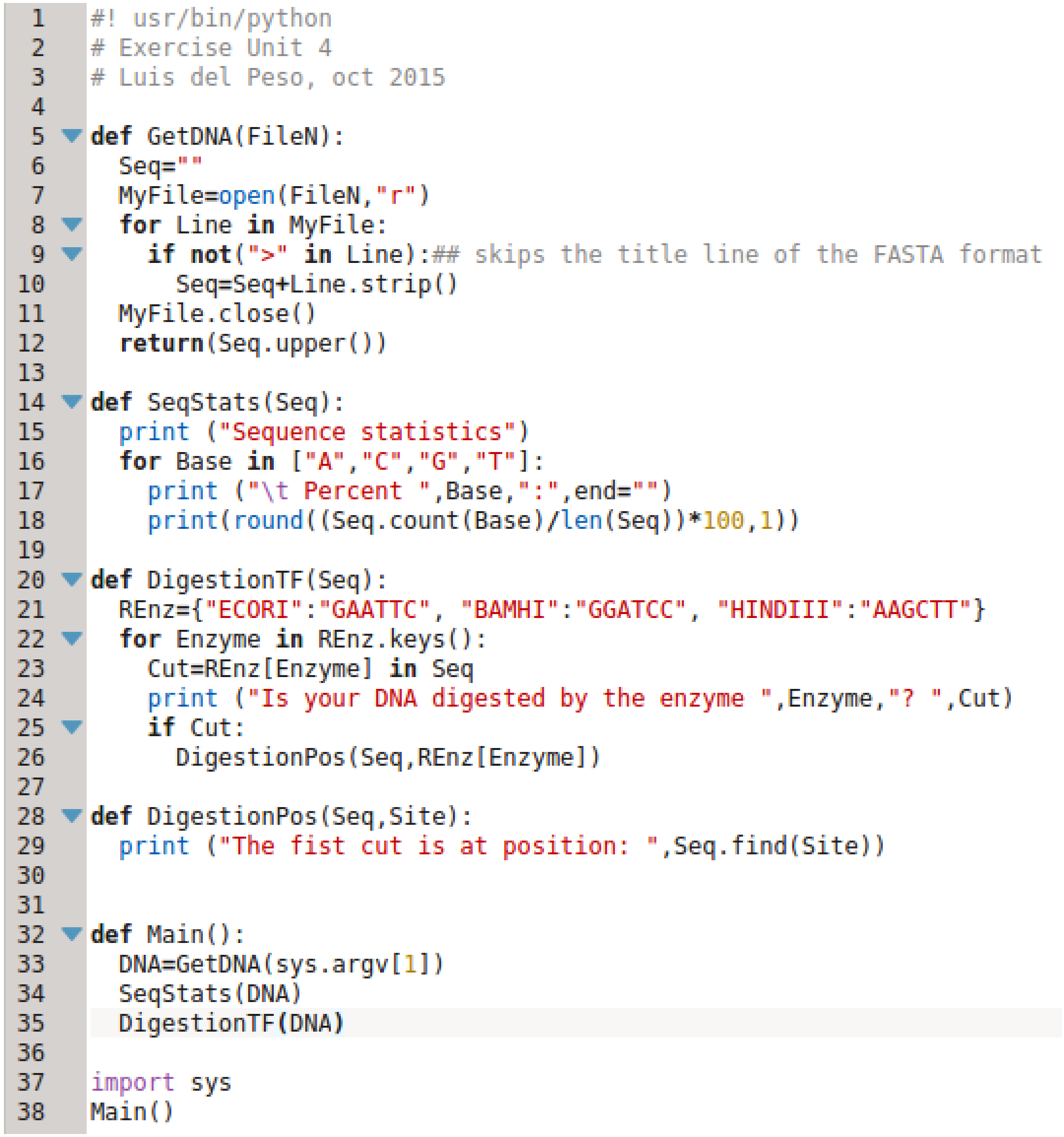
Variables defined within functions are local variables and their *scope* is within that function.

e.g. variables *NumberList*, *suma* and *result* are *local* variables whose *scope* is within the function *Media*. They can be accesed outside this function.

Variables defined within the main program are global variables and can be accessed from any part of program or from within any function.

e.g. *Weight4Bq* and *AverageWeight4Bq* are *local* variables whose *scope* is all program

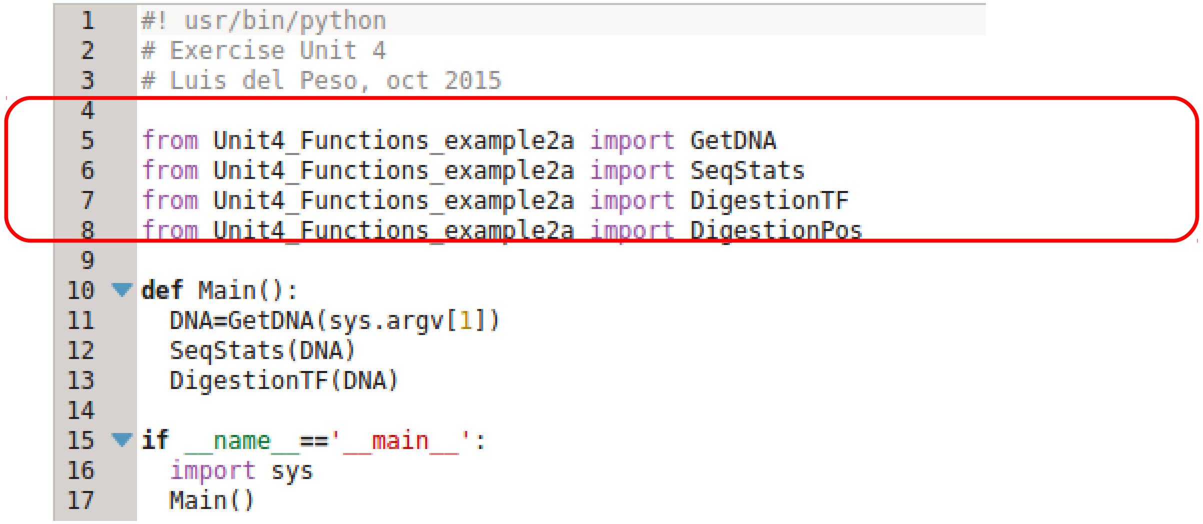
Program Structure: modularity (I)



```
#! usr/bin/python
# Exercise Unit 4
# Luis del Peso, oct 2015

def GetDNA(FileN):
  Seq=""
  MyFile=open(FileN,"r")
  for Line in MyFile:
    if not(">" in Line):## skips the title line of the FASTA format
      Seq=Seq+Line.strip()
  MyFile.close()
  return(Seq.upper())

def SeqStats(Seq):
  print ("Sequence statistics")
  for Base in ["A","C","G","T"]:
    print ("\t Percent ",Base,":",end="")
    print(round((Seq.count(Base)/len(Seq))*100,1))

def DigestionTF(Seq):
  REnz={"ECORI":"GAATTC", "BAMHI":"GGATCC", "HINDIII":"AAGCTT"}
  for Enzyme in REnz.keys():
    Cut=REnz[Enzyme] in Seq
    print ("Is your DNA digested by the enzyme ",Enzyme,"? ",Cut)
    if Cut:
      DigestionPos(Seq,REnz[Enzyme])

def DigestionPos(Seq,Site):
  print ("The fist cut is at position: ",Seq.find(Site))


def Main():
  DNA=GetDNA(sys.argv[1])
  SeqStats(DNA)
  DigestionTF(DNA)

import sys
Main()
```

Unit4_Functions_example2a.py
Unit4_Functions_example2b.py

Program Structure: Recycling code



```python
#! usr/bin/python
# Exercise Unit 4
# Luis del Peso, oct 2015

from Unit4_Functions_example2a import GetDNA
from Unit4_Functions_example2a import SeqStats
from Unit4_Functions_example2a import DigestionTF
from Unit4_Functions_example2a import DigestionPos

def Main():
  DNA=GetDNA(sys.argv[1])
  SeqStats(DNA)
  DigestionTF(DNA)

if __name__=='__main__':
  import sys
  Main()
```

Note that we are using functions from another program a

Unit4_Functions_example3.py