# INFORME PROGRAMAS DE ORDENAMIENTO EN PYTHON CON RESULTADOS

# Heapsort

#-------------------------------------------------------------------------------

# Name: HeapSort

#

# Author: Ivan Mansalva

#

# Created: 03/10/2012

# Copyright: (c) Ivalejo 2012

# Licence: <your licence>

#-------------------------------------------------------------------------------

import time

def HeapSort(A):

def heapify(A):

start = (len(A) - 2) / 2

while start >= 0:

siftDown(A, start, len(A) - 1)

start -= 1

def siftDown(A, start, end):

root = start

while root \* 2 + 1 <= end:

child = root \* 2 + 1

if child + 1 <= end and A[child] < A[child + 1]:

child += 1

if child <= end and A[root] < A[child]:

A[root], A[child] = A[child], A[root]

root = child

else:

return

heapify(A)

end = len(A) - 1

while end > 0:

A[end], A[0] = A[0], A[end]

siftDown(A, 0, end - 1)

end -= 1

if \_\_name\_\_ == '\_\_main\_\_':

print "Tiempo inicio:"

t\_inicio = time.clock()

T = [13, 14, 94, 33,100, 82, 25, 59, 94, 65, 23, 45, 27, 73, 25, 39, 10]

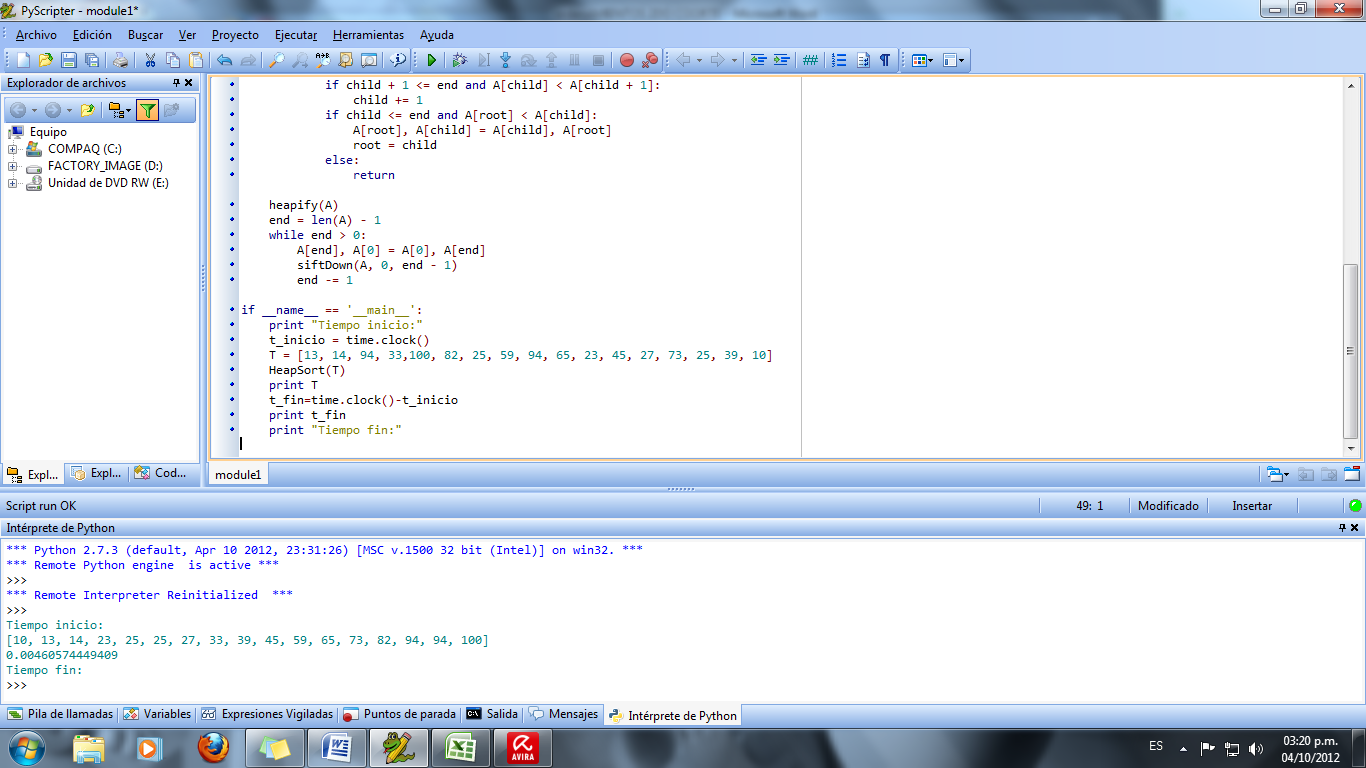
HeapSort(T)

print T

t\_fin=time.clock()-t\_inicio

print t\_fin

print "Tiempo fin:"



>>>

Tiempo inicio:

[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 7]

0.000163816371291

Tiempo fin

>>>

# Quicksort

#-------------------------------------------------------------------------------

# Name: QuickSort

# Purpose:

#

# Author: Ivan Mansalva

#

# Created: 03/10/2012

# Copyright: (c) Ivalejo 2012

# Licence: <your licence>

#-------------------------------------------------------------------------------

import time

def main():

pass

if \_\_name\_\_ == '\_\_main\_\_':

def quickSort(vector):

if len(vector) <= 1:

return vector

end = len(vector) - 1

pivot = vector[end]

low = []

high = []

for num in vector[:end]:

if num <= pivot:

low.append(num)

else:

high.append(num)

listaOrdenada = quickSort(low)

listaOrdenada.append(pivot)

listaOrdenada.extend(quickSort(high))

return listaOrdenada

def main():

t1=time.clock()

print("Tiempo Inicio Ejecucion : ",t1)

l = [1, 6, 7, 2, 76, 45, 23, 4, 8, 12, 11]

listaOrdenada = quickSort(l)

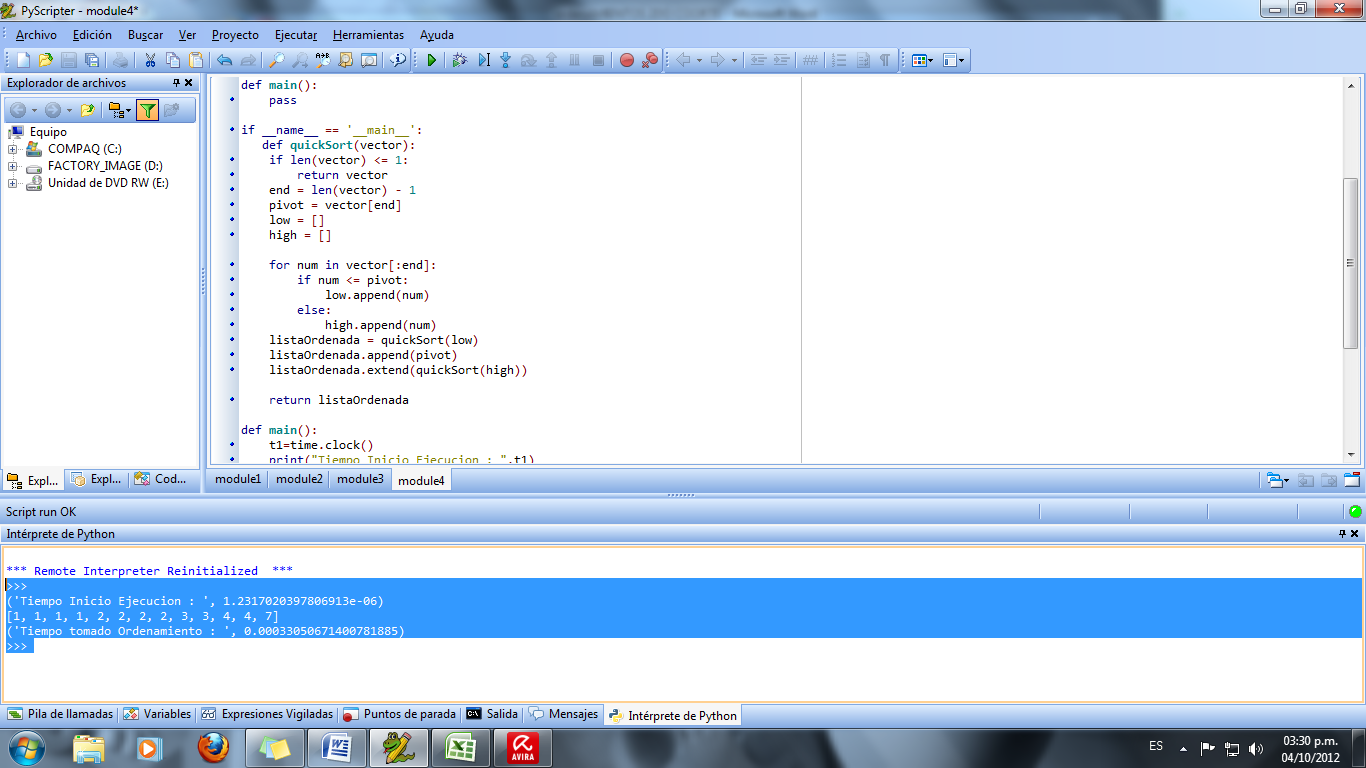
print listaOrdenada

t2=time.clock()

print("Tiempo tomado Ordenamiento : ",(t2-t1))

if \_\_name\_\_ == '\_\_main\_\_':

main()



>>>

('Tiempo Inicio Ejecucion : ', 1.2317020397806913e-06)

[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 7]

('Tiempo tomado Ordenamiento : ', 0.00033050671400781885)

>>>

# CountingSort

#-------------------------------------------------------------------------------

# Name: QuickSort

# Purpose:

#

# Author: Ivan Mansalva

# Created: 03/10/2012

# Copyright: (c) Ivalejo 2012

# Licence: <your licence>

#-------------------------------------------------------------------------------

import time

def counting\_sort(array, maxval):

"""in-place counting sort"""

t1=time.clock()

print("Tiempo Inicio Ejecucion : ",t1)

m = maxval + 1

count = [0] \* m # init with zeros

for a in array:

count[a] += 1 # count occurences

i = 0

for a in range(m): # emit

for c in range(count[a]): # - emit 'count[a]' copies of 'a'

array[i] = a

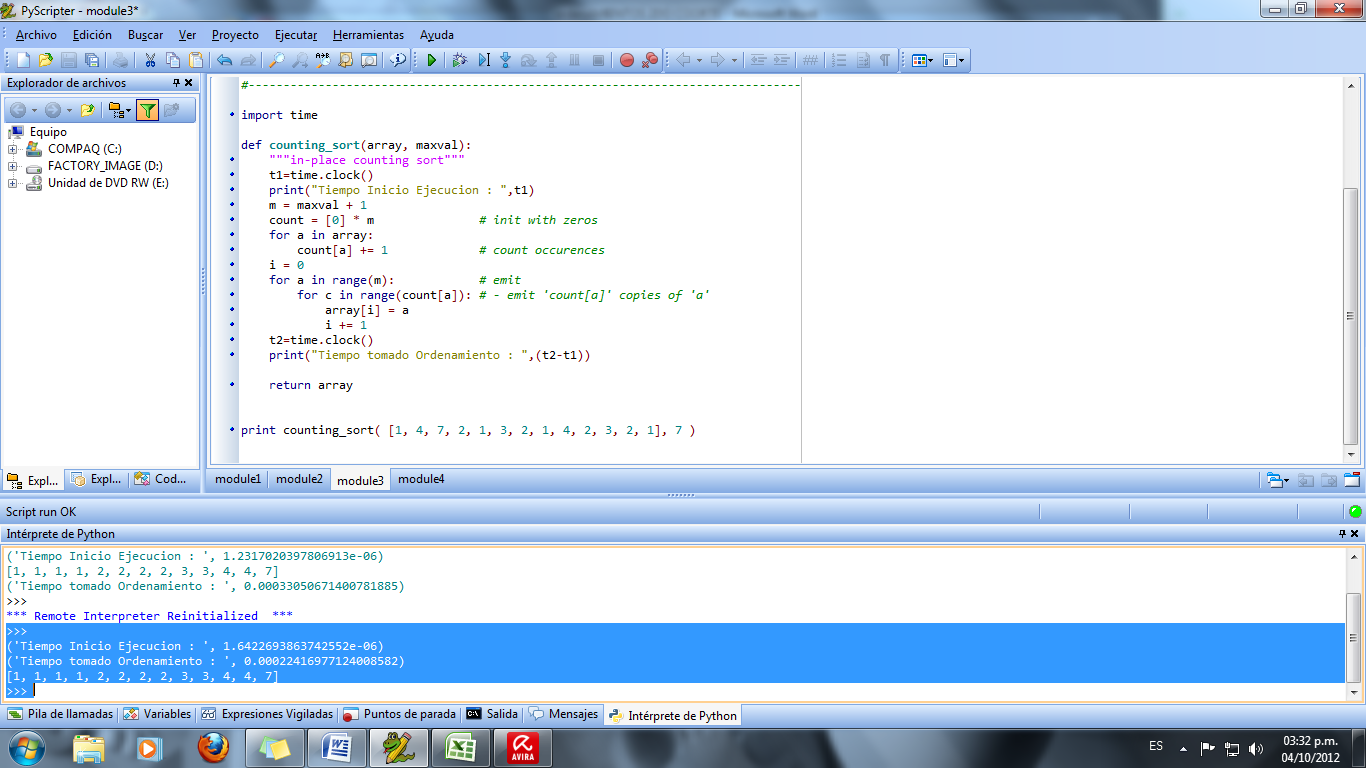
i += 1

t2=time.clock()

print("Tiempo tomado Ordenamiento : ",(t2-t1))

return array

print counting\_sort( [1, 4, 7, 2, 1, 3, 2, 1, 4, 2, 3, 2, 1], 7 )



>>>

('Tiempo Inicio Ejecucion : ', 1.6422693863742552e-06)

('Tiempo tomado Ordenamiento : ', 0.00022416977124008582)

[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 7]

>>>

# RadixSort

#-------------------------------------------------------------------------------

# Name: Radixsort

# Purpose:

#

# Author: Ivan Mansalva

#

# Created: 03/10/2012

# Copyright: (c) Ivalejo 2012

# Licence: <your licence>

#-------------------------------------------------------------------------------

# Radix sort for numbers in base 10

import math

import time

def radixSortNumbers(array):

t1=time.clock()

print("Tiempo Inicio Ejecucion : ",t1)

maxLen = 10

for number in array: # Find longest number, in digits

numLen = int(math.log10(number)) + 1

if numLen > maxLen:

maxLen = numLen

buckets = [[] for i in range(0, 10)] # Buckets for each digit

for digit in range(0, maxLen):

for number in array:

buckets[number / 10\*\*digit % 10].append(number)

del array[:]

for bucket in buckets:

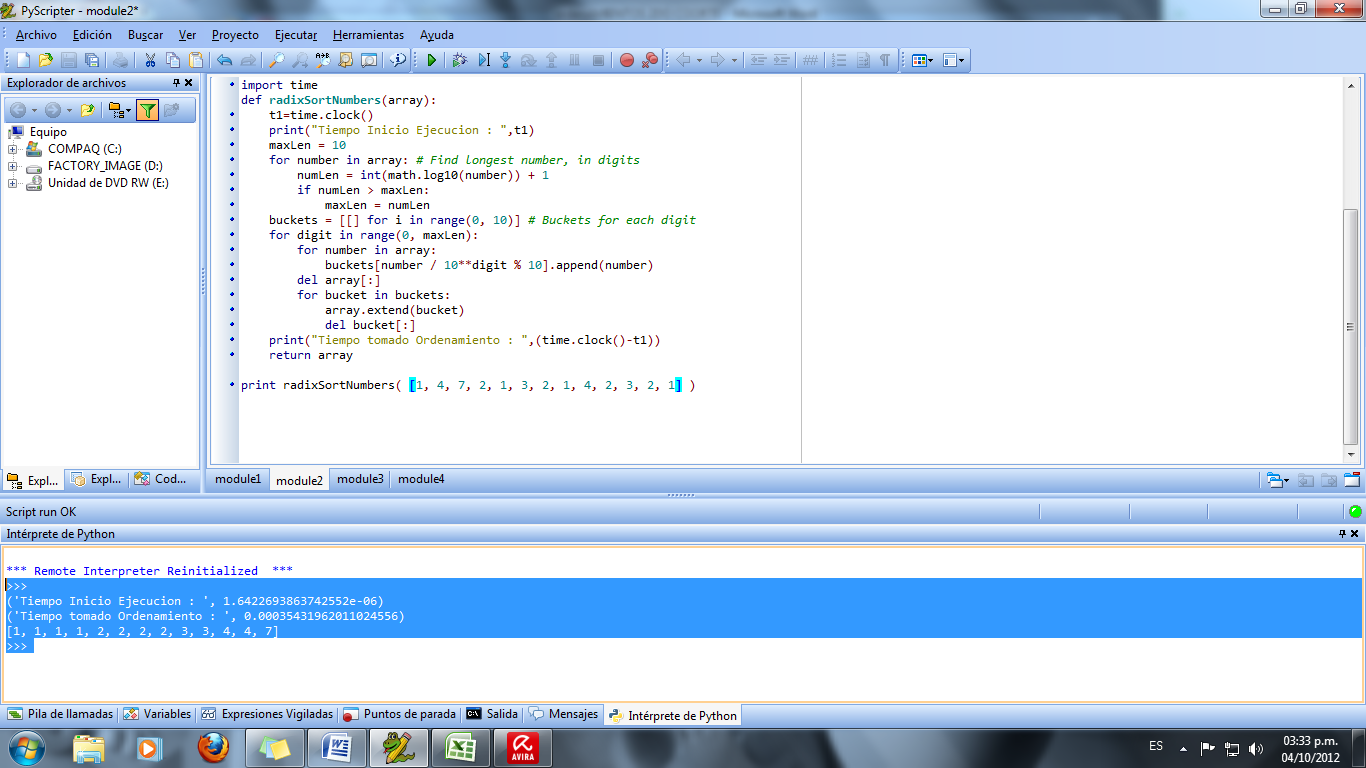
array.extend(bucket)

del bucket[:]

print("Tiempo tomado Ordenamiento : ",(time.clock()-t1))

return array

print radixSortNumbers( [1, 4, 7, 2, 1, 3, 2, 1, 4,8] )



>>>

('Tiempo Inicio Ejecucion : ', 1.6422693863742552e-06)

('Tiempo tomado Ordenamiento : ', 0.00035431962011024556)

[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 7]

>>>

|  |  |  |  |
| --- | --- | --- | --- |
| Tiempo de resultados con diferentes ordenamientos y cantidad de elementos en el arreglo | | | |
|  | Tiempo 13 elementos | Tiempo 7 elementos | Tiempo 5 elementos |
| Tipos de Ordenamientos | Milisegundos | | |
| **RadixSort** | 0.00035431962011024556 | 0.00024880381203569966 | 0.00024798267734251253 |
| **CountingSort** | 0.00022416977124008582 | 0.0001744911223022646 | 0.00021144218349568533 |
| **Quicksort** | 0.00033050671400781885 | 0.00030258813443945654 | 0.0003210636650361669 |
| **Heapsort** | 0.000163816371291 | 0.000152320485586 | 0.000159710697825 |