

Tarea 2

Unidad1: Algoritmos de Ordenamiento.

Profesor. Gerardo Tovar Tapia.

Alumna. Monroy Velázquez Alejandra Sarahí

- 1) Del libro “python para informáticos”. Leer el capítulo 2, y capítulo 3 y hacer los ejercicios que se proponen en de dichas sección. Además programar los ejemplos que se van dando en los capítulos.

Ejercicios: Capitulo 2

Ejercicio 2.2 Escribe un programa que use raw_input para pedirle al usuario su nombre y luego darle la bienvenida.

Introduzca tu nombre: Chuck

Hola, Chuck

Python 2.7

```
1 nombre = raw_input("¿Cual es tu nombre? ")
→ 2 print("Hola, "+ nombre)
```

Ejercicio2.2.py

Print output (drag lower right corner to resize)

```
¿Cual es tu nombre? Alejandra
Hola, Alejandra
```

Es importante mencionar que el `raw_input()` solo funciona en la versión 2 de python, en la versión actual(3), se usa `input()` para ingresar los datos, ya que al utilizar `raw_input()` muestra un error. Lo correcto es:



```
1 nombre = input("¿Cual es tu nombre?")
2 print("Hola, ", nombre)
```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux

```
> ¿Cual es tu nombre? Alejandra
> Hola, Alejandra
```

Ejercicio 2.3 Escribe un programa para pedirle al usuario el número de horas y la tarifa por hora para calcular el salario bruto.

Introduzca Horas: 35

Introduzca Tarifa: 2.75

Salario: 96.25

Por ahora no es necesario preocuparse de que nuestro salario tenga exactamente dos dígitos después del punto decimal. Si quieres, puedes probar la función interna de Python round para redondear de forma adecuada el salario resultante a dos dígitos decimales.

```

1 horas = float(input("Introduzca horas: "))
2 tarifa = float(input("Introduzca tarifa: "))
3 salario = horas * tarifa
4 print("Salario = ",salario)
5

```

```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Introduzca horas: 35
Introduzca tarifa: 2.75
Salario = 96.25
>

```

Ejercicio2.3.py

Ejercicio 2.4 Asume que ejecutamos las siguientes sentencias de asignación:

ancho = 17

alto = 12.0

Para cada una de las expresiones siguientes, escribe el valor de la expresión y el tipo (del valor de la expresión).

1. *ancho/2 – 8.5*

2. *ancho/2.0 – 8.5*

3. *alto/3 - 4.0*

4. *1 + 2 * 5 - 11*

Usa el intérprete de Python para comprobar tus respuestas.

Ejercicio 2.5 Escribe un programa que le pida al usuario una temperatura en grados Celsius, la convierta a grados Fahrenheit e imprima por pantalla la temperatura convertida.

```

1 tempc = float(input("Introduce la temperatura en grados Celsius: "))
2 tempf = tempc *(9/5) +32
3 print("La temperatura en Farenheit es: ",tempf)
4
5

```

```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Introduce la temperatura en grados Celsius: 8
La temperatura en Farenheit es: 46.4
>

```

Ejercicio2.5.py

Ejercicios: Capitulo 3

Ejercicio 3.1 Reescribe el programa del cálculo del salario para darle al empleado 1.5 veces la tarifa horaria para todas las horas trabajadas que excedan de 40.

Introduzca las Horas: 40

Introduzca la Tarifa por hora: 10

Salario: 475.0

```

1 horas = float(input("Introduzca las horas: "))
2 tarifa = float(input("Introduzca la tarifa por hora: "))
3 if horas >= 40:
4     vecesh = horas * 1.5
5     vecest = tarifa * 1.5
6     salario = (horas * tarifa) + (vecesh + vecest)
7     print("Salario = ",salario)
8 else:
9     salario = horas * tarifa
10    print("Salario = ",salario)
11

```

```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Introduzca las horas: 40
Introduzca la tarifa por hora: 10
Salario = 475.0
>

```

Ejercicio3.1.py

Ejercicio 3.2 Reescribe el programa del salario usando try y except, de modo que el programa sea capaz de gestionar entradas no numéricas con elegancia, mostrando un mensaje y saliendo del programa. A continuación se muestran dos ejecuciones del programa:

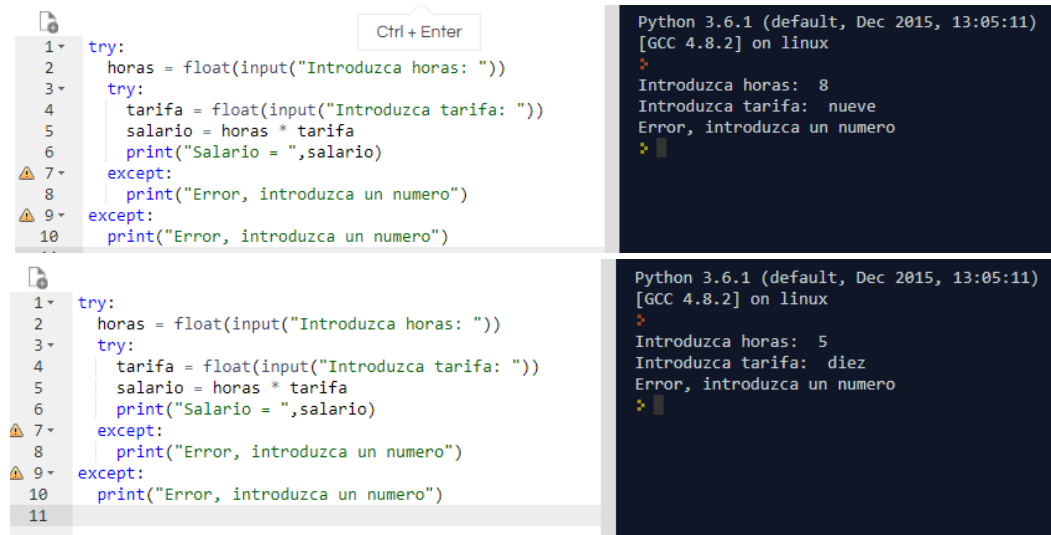
Introduzca las Horas: 20

Introduzca la Tarifa por hora: nueve

Error, por favor introduzca un número

Introduzca las Horas: cuarenta

Error, por favor introduzca un número



```
1 try:
2     horas = float(input("Introduzca horas: "))
3     try:
4         tarifa = float(input("Introduzca tarifa: "))
5         salario = horas * tarifa
6         print("Salario = ",salario)
7     except:
8         print("Error, introduzca un numero")
9 except:
10    print("Error, introduzca un numero")
11
```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Introduzca horas: 8
Introduzca tarifa: nueve
Error, introduzca un numero

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Introduzca horas: 5
Introduzca tarifa: diez
Error, introduzca un numero

Ejercicio3.2.py

Ejercicio 3.3 Escribe un programa que solicite una puntuación entre 0.0 y 1.0. Si la puntuación está fuera de ese rango, muestra un mensaje de error. Si la puntuación está entre 0.0 y 1.0, muestra la calificación usando la tabla siguiente:

Puntuación Calificación

>= 0.9 Sobresaliente

>= 0.8 Notable

>= 0.7 Bien

>= 0.6 Suficiente

< 0.6 Insuficiente

Introduzca puntuación: 0.95

Sobresaliente

Introduzca puntuación: perfecto

Puntuación incorrecta

Introduzca puntuación: 10.0

Puntuación incorrecta

Introduzca puntuación: 0.75

Bien

Introduzca puntuación: 0.5

Insuficiente

Ejecuta el programa repetidamente, como se muestra arriba, para probar con varios valores de entrada diferentes.

```
1 try:
2     puntuacion = float(input("Ingrese la puntuación entre 0 y 1: "))
3     if puntuacion > 1:
4         print("Puntuacion incorrecta")
5     else:
6         if puntuacion >= 0.9:
7             print("Sobresaliente")
8         elif puntuacion >= 0.8:
9             print("Notable")
10        elif puntuacion >= 0.7:
11            print("Bien")
12        elif puntuacion >= 0.6:
13            print("Suficiente")
14        elif puntuacion < 0.6:
15            print("Insuficiente")
16 except:
17     print("Puntuacion incorrecta")
18
19
20
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Ingrese la puntuación entre 0 y 1: 0.95
>
Ingrese la puntuación entre 0 y 1: perfecto
Puntuacion incorrecta
>
Ingrese la puntuación entre 0 y 1: 10.0
Puntuacion incorrecta
>
Ingrese la puntuación entre 0 y 1: 0.75
Bien
>
Ingrese la puntuación entre 0 y 1: 0.5
Insuficiente
>
```

Ejercicio3.3.py

2) Realice los siguientes programas.

- Desarrolle un algoritmo que genere los 300 números enteros y determine cuántos de ellos son primos, y cuales son pares; al final deberá indicar su sumatoria.

```
1 def par(i):
2     if(i%2 == 0):
3         print("Es par:")
4
5 def primo(i):
6     a=0
7     for j in range(1,i+1):
8         if(i%j==0):
9             a=a+1
10    if(a==2):
11        print("Es primo:")
12
13
14 for i in range(300):
15     print (i+1)
16     primo(i)
17     par(i)
18
```

```
282
Es primo:
283
Es par:
284
Es primo:
285
Es par:
286
287
Es par:
288
289
Es par:
290
291
Es par:
292
293
```

PrimosYPares.py

- Definir una función max() que tome como argumento dos números y devuelva el mayor de ellos. (Es cierto que python tiene una función max() incorporada, pero hacerla nosotros mismos).

```

1 def max(a,b):
2     while a == b:
3         print("Son iguales, vuelve a intentar")
4         a= int(input("Primer numero: "))
5         b= int(input("Segundo numero: "))
6
7     if a > b:
8         print("El numero mayor es:",a)
9
10    else:
11        print("El numero mayor es:",b)
12
13    print("Introduce dos numeros")
14    a= int(input("Primer numero: "))
15    b= int(input("Segundo numero: "))
16    max(a,b)
17
18
19

```

defmax.py

```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Introduce dos numeros
Primer numero: 8
Segundo numero: 8
Son iguales, vuelve a intentar
Primer numero: 2
Segundo numero: 1
El numero mayor es: 2
>

```

- Definir una función max_de_tres(), que tome tres números como argumentos y devuelva el mayor de ellos.

```

1 def max_de_tres(a,b,c):
2     if a > b and a > c:
3         print("El numero mayor es:",a)
4
5     elif b > a and b > c:
6         print("El numero mayor es:",b)
7
8     else:
9         print("El numero mayor es:",c)
10
11
12    a= int(input("Primer numero: "))
13    b= int(input("Segundo numero: "))
14    c= int(input("Tercer numero: "))
15    max_de_tres(a,b,c)
16
17

```

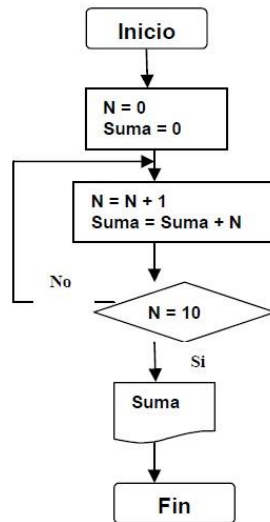
defmaxdetres.py

```

Python 3.6.1 (default, Dec 2015, 13:
[GCC 4.8.2] on linux
>
Primer numero: 4
Segundo numero: 9
Tercer numero: 7
El numero mayor es: 9
>

```

3. Programar los diagramas de flujo siguientes
a)

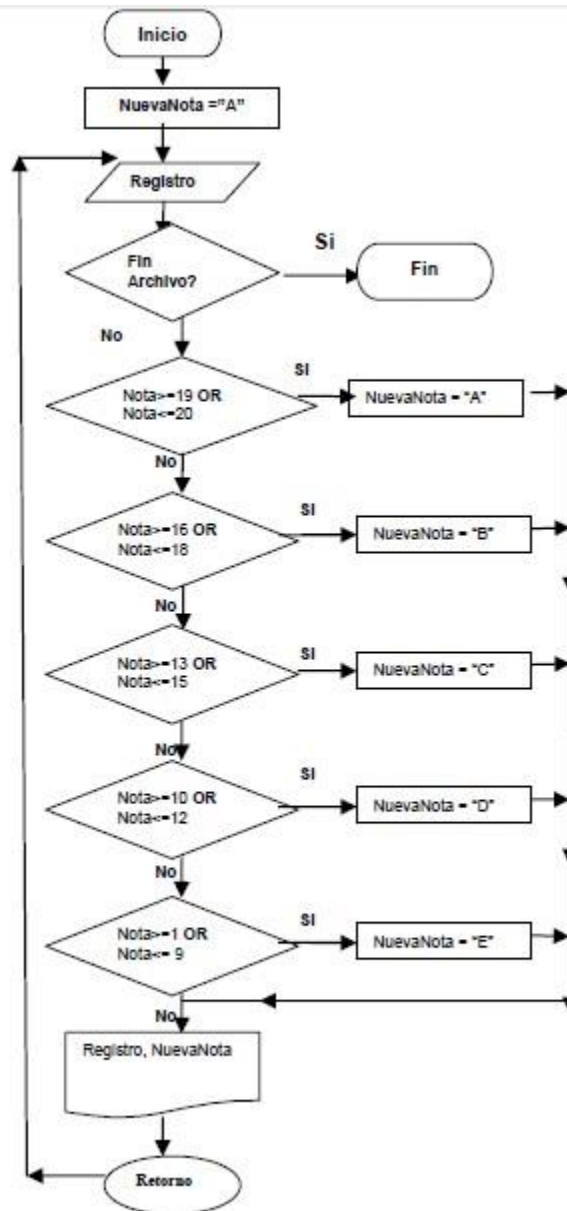


```
1 n=0
2 suma=0
3 for i in range(10):
4     n=n+1
5     suma = suma + n
6
7 print("La suma es: ",suma)
8
```

[GCC 4.8.2] on linux

```
La suma es: 55
```

diagrama1.py



b)

```

1- def main():
2-     op=int(input("Desea hacer un registro? \nPresione 1 para hacerlo,
3-     presione cualquier tecla para salir: "))
4-     if (op == 1):
5-         nota=int(input("Ingrese Nota: "))
6-         if(nota>=19 and nota<=20):
7-             print("Nueva nota = A")
8-             main()
9-         elif(nota>=16 and nota<=18):
10-            print("Nueva nota = B")
11-            main()
12-        elif(nota>=13 and nota<=15):
13-            print("Nueva nota = C")
14-            main()
15-        elif(nota>=10 and nota<=12):
16-            print("Nueva nota = D")
17-            main()
18-        elif(nota>=1 and nota<=9):
19-            print("Nueva nota = E")
20-            main()
21-    main()
22-

```

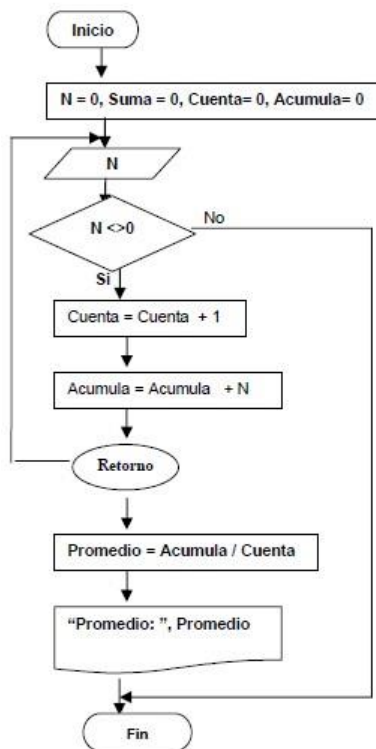
[GCC 4.8.2] on linux

```

>
Desea hacer un registro?
Presione 1 para hacerlo, presione cualquier tecla para salir: 1
Ingrese Nota: 4
Nueva nota = E
Desea hacer un registro?
Presione 1 para hacerlo, presione cualquier tecla para salir: 1
Ingrese Nota: 20
Nueva nota = A
Desea hacer un registro?
Presione 1 para hacerlo, presione cualquier tecla para salir: 1
Ingrese Nota: 11
Nueva nota = D
Desea hacer un registro?
Presione 1 para hacerlo, presione cualquier tecla para salir: 1
Ingrese Nota: 14
Nueva nota = C
Desea hacer un registro?
Presione 1 para hacerlo, presione cualquier tecla para salir: 1
Ingrese Nota: 18
Nueva nota = B
Desea hacer un registro?
Presione 1 para hacerlo, presione cualquier tecla para salir: 2
>

```

c)



```

1 def main():
2     n=0
3     suma=0
4     cuenta=0
5     acumula=0
6
7     op=input("Desea ingresar calificacion (s/n): ")
8
9     while (op != 'n'):
10        n=float(input("Ingrese calificacion: "))
11        cuenta=cuenta+1
12        acumula=acumula+n
13        op=input("Desea ingresar calificacion (s/n): ")
14        pass
15    promedio=acumula/cuenta
16    print("El promedio es: {0} ".format(promedio))
17
18    main()
19

```

```

[GCC 4.8.2] on linux
Desea ingresar calificacion (s/n): s
Ingrese calificacion: 5
Desea ingresar calificacion (s/n): s
Ingrese calificacion: 7
Desea ingresar calificacion (s/n): s
Ingrese calificacion: 9
Desea ingresar calificacion (s/n): s
Ingrese calificacion: 2
Desea ingresar calificacion (s/n): n
El promedio es: 5.75

```

diagrama3.py

4. Del libro “Introducción al Análisis y diseño de Algoritmos” de la Dra. María del Carmen Gómez Fuentes, y el Dr. Jorge Cervantes Ojeda, que se adjunta. Leer la introducción “Eficiencia y métricas de un algoritmo” y hacer un breve resumen con los puntos que usted crea más importantes.

Un algoritmo es un conjunto de pasos que, ejecutados de la manera correcta, permiten obtener un resultado en un tiempo acotado. Un algoritmo tiene las siguientes propiedades:

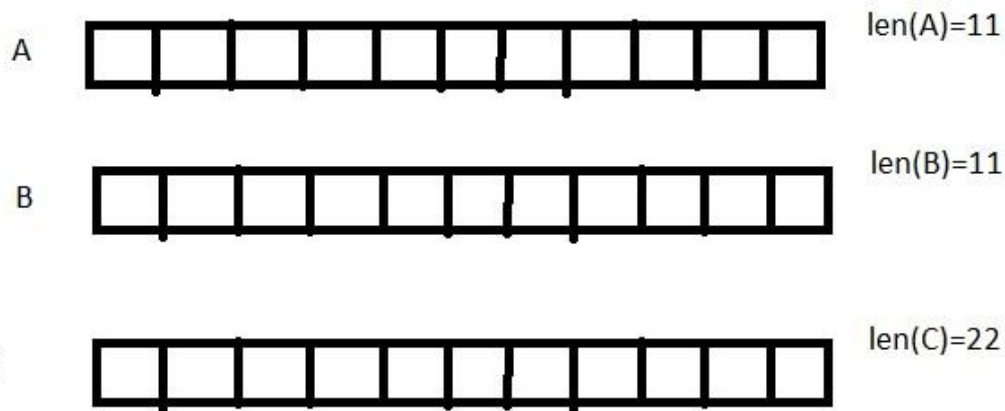
- Entrada y Salida
- Preciso
- Determinista
- Finito
- Efectivo

El análisis de algoritmos pretende descubrir si estos son o no eficaces. Es responsabilidad del programador utilizar los recursos de la computadora de la manera más eficiente. El tiempo de ejecución de un algoritmo depende de los datos de entrada, de la implementación del programa, del procesador, y finalmente de la complejidad del algoritmo. Al tiempo que tarda un algoritmo para resolver un problema se le llama $T(n)$, donde n es el tamaño de los datos de entrada. Es decir, $T(n)$ depende del tamaño de los datos de entrada. Para medir $T(n)$ usamos el número de operaciones elementales. Una operación elemental puede ser:

- Operación aritmética.
- Asignación a una variable.
- Llamada a una función.
- Retorno de una función.
- Comparaciones lógicas (con salto).
- Acceso a una estructura (arreglo, matriz, lista ligada...).

Se le llama tiempo de ejecución, no al tiempo físico, sino al número de operaciones elementales que se llevan a cabo en el algoritmo.

5. Se tienen 2 arreglos de tamaño 11 cada uno, y que contienen elementos que no se repiten, hacer un programa que lea y ordene en el arreglo C a los arreglos A , y B , además se deberá imprimir el tiempo que tarda el programa en realizar dicha tarea.



```

1 from time import clock
2 def CrearSubArreglo(listac, indIzq, indDer):
3     return listac[indIzq:indDer+1]
4
5 def Merge(listac,p,q,r):
6     Izq = CrearSubArreglo(listac,p,q)
7     Der = CrearSubArreglo(listac,q+1,r)
8     i = 0
9     j = 0
10    for k in range(p,r+1):
11        if (j >= len(Der)) or (i < len(Izq) and Izq[i] < Der[j]):
12            listac[k] = Izq[i]
13            i = i + 1
14        else:
15            listac[k] = Der[j]
16            j = j + 1
17
18 def MergeSort(listac,p,r):
19     if r - p > 0:
20         q = int((p+r)/2)
21         MergeSort(listac,p,q)
22         MergeSort(listac,q+1,r)
23         Merge(listac,p,q,r)
24
25 listaA = []
26 listaB = []
27 listaC = []
28 for i in range(11):
29     n = int(input("Inserte un elemento: "))
30     listaA.append(n)
31 print("La lista 1 contiene los siguientes elementos: ",listaA)
32 for i in range(11):
33     n = int(input("Inserte un elemento: "))
34     listaB.append(n)
35 print("La lista 2 contiene los siguientes elementos: ",listaB)
36 listaC = listaA + listaB
37 print("La nueva lista contiene los siguientes elementos: ",listaC)
38 tiempoi = clock()
39 MergeSort(listaC, 0, 21)
40 tiempof = clock()
41 print("La lista ordenada es: ",listaC)
42 print("Tiempo de ejecución: ",(tiempof-tiempoi))

```

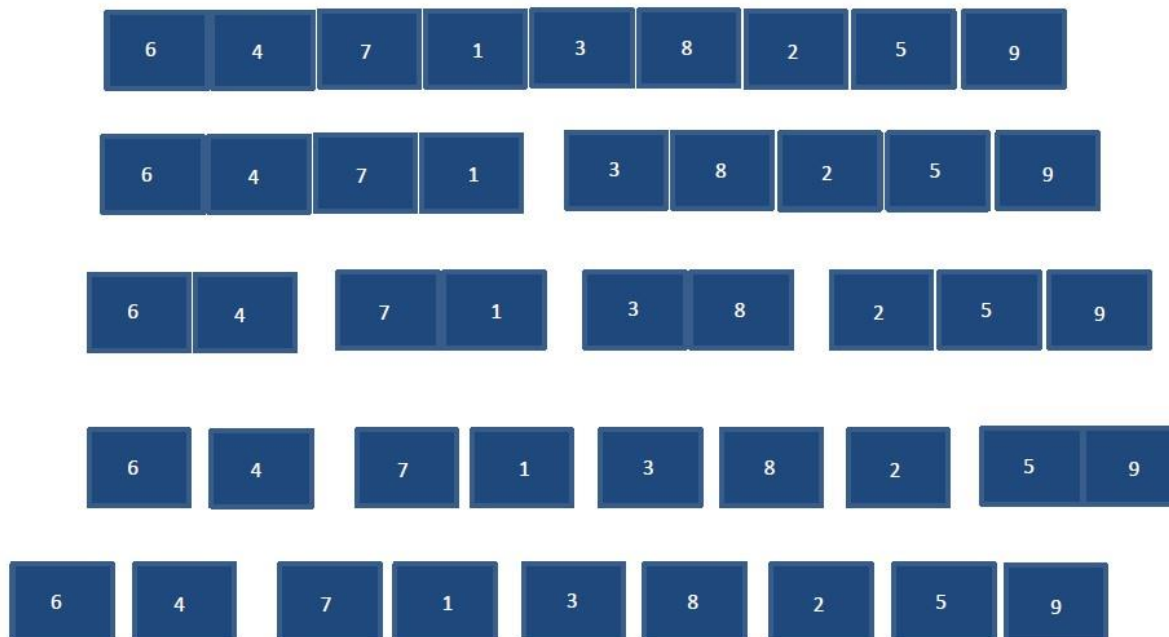
```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
Inserte un elemento: 22
Inserte un elemento: 21
Inserte un elemento: 20
Inserte un elemento: 19
Inserte un elemento: 18
Inserte un elemento: 17
Inserte un elemento: 16
Inserte un elemento: 15
Inserte un elemento: 14
Inserte un elemento: 13
Inserte un elemento: 12
La lista 1 contiene los siguientes elementos: [22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12]
Inserte un elemento: 11
Inserte un elemento: 10
Inserte un elemento: 9
Inserte un elemento: 8
Inserte un elemento: 7
Inserte un elemento: 6
Inserte un elemento: 5
Inserte un elemento: 4
Inserte un elemento: 3
Inserte un elemento: 2
Inserte un elemento: 1
La lista 2 contiene los siguientes elementos: [11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
La nueva lista contiene los siguientes elementos: [22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
La lista ordenada es: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]
Tiempo de ejecución: 9.499999999999786e-05
>

```

OrdenarListaC.py

6. Hacer un programa utilizando la técnica “Divide y vencerás”, que pasándole un arreglo de tamaño N, imprima todas las segmentaciones de dicho arreglo, tal y como se muestra a continuación, hasta llegar a arreglos de tamaño 1.



```

1 def Segmentar(listaA,p,r):
2     if r - p > 0:
3         q = int((p+r)/2)
4         print("-")
5         print(listaA[p:r+1])
6         if(len(listaA[p:r+1]) == 2):
7             print("[",listaA[p],"]")
8             print("[",listaA[r],"]")
9             Segmentar(listaA,p,q)
10            Segmentar(listaA,q+1,r)
11
12
13 listaA = [6,4,7,1,3,8,2,5]
14 print("La lista contiene los siguientes elementos: ",listaA)
15 Segmentar(listaA, 0, 7)
16
17
18

```

```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
La lista contiene los siguientes elementos: [6, 4, 7, 1, 3, 8, 2, 5]
-
[6, 4, 7, 1, 3, 8, 2, 5]
-
[6, 4, 7, 1]
-
[6, 4]
[ 6 ]
[ 4 ]
-
[7, 1]
[ 7 ]
[ 1 ]
-
[3, 8, 2, 5]
-
[3, 8]
[ 3 ]
[ 8 ]
-
[2, 5]
[ 2 ]
[ 5 ]
>

```

Segmentaciones.py

Nota: Todos los programas vienen en el zip "grupo6_EDA_ProgramasTarea2"