

Tarea 3

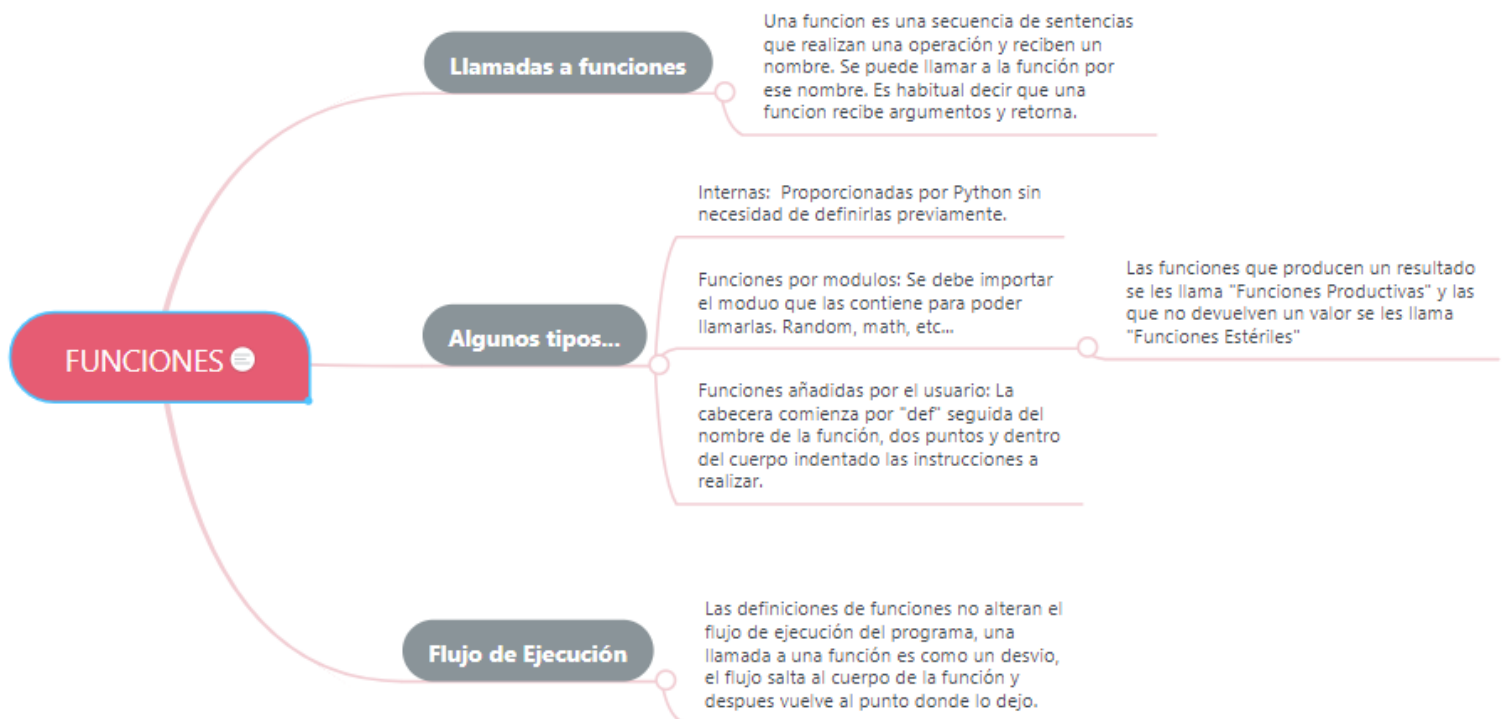
Unidad1: Algoritmos de Ordenamiento.

Profesor. Gerardo Tovar Tapia.

Alumna: Monroy Velázquez Alejandra Sarahí

- 1) Del libro **Python para informáticos, Charles Severance**. Leer el capítulo 4 y 5, y hacer un cuadro sinóptico con los puntos más relevantes. Y resolver los ejercicios de fin de sección de cada capítulo.

CAPITULO 4



Ejercicio 4.1 Ejecuta el programa en tu sistema y observa qué números obtienes.

```
1 import random
2
3 for i in range(10):
4     x= random.random()
5     print(x)
6
```

[GCC 4.8.2] on linux

0.3317702763786827
0.5557570134914479
0.9796861323969167
0.7258322831416665
0.03863439684145198
0.3440480378140428
0.5472747914959647
0.6443888015262378
0.8541101356409876
0.22967014658462526

Ejercicio 4.2 Desplaza la última línea de este programa hacia arriba, de modo que la llamada a la función aparezca antes que las definiciones. Ejecuta el programa y observa qué mensaje de error obtienes.

```
1 repite_estribillo()
2
3 def muestra_estribillo():
4     print ('Soy un leñador, que alegría.')
5     print ('Duermo toda la noche y trabajo todo el día.')
6
7 def repite_estribillo():
8     muestra_estribillo()
9     muestra_estribillo()
10
11
```

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

Traceback (most recent call last):
File "python", line 1, in <module>
NameError: name 'repite_estribillo' is not defined

La función no está definida para poder llamarla.

Ejercicio 4.3 Desplaza la llamada de la función de nuevo hacia el final, y coloca la definición de muestra_estribillo después de la definición de repite_estribillo. ¿Qué ocurre cuando haces funcionar ese programa?

```
1 def repite_estribillo():
2     muestra_estribillo()
3     muestra_estribillo()
4
5 def muestra_estribillo():
6     print ('Soy un leñador, que alegría.')
7     print ('Duermo toda la noche y trabajo todo el día.')
8
9 repite_estribillo()
```

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

Soy un leñador, que alegría.
Duermo toda la noche y trabajo todo el día.
Soy un leñador, que alegría.
Duermo toda la noche y trabajo todo el día.

Lo ejecuta normalmente.

Ejercicio 4.4 ¿Cuál es la utilidad de la palabra clave “def” en Python?

- a) Es una jerga que significa “este código es realmente estupendo”
- b) Indica el comienzo de una función
- c) Indica que la siguiente sección de código indentado debe ser almacenada para usarla más tarde
- d) b y c son correctas ambas
- e) Ninguna de las anteriores

Ejercicio 4.5 ¿Qué mostrará en pantalla en siguiente programa Python?

```
def fred():  
    print "Zap"  
def jane():  
    print "ABC"  
jane()  
fred()  
jane()
```

- a) Zap ABC jane fred jane
- b) Zap ABC Zap
- c) ABC Zap jane
- d) ABC Zap ABC
- e) Zap Zap Zap

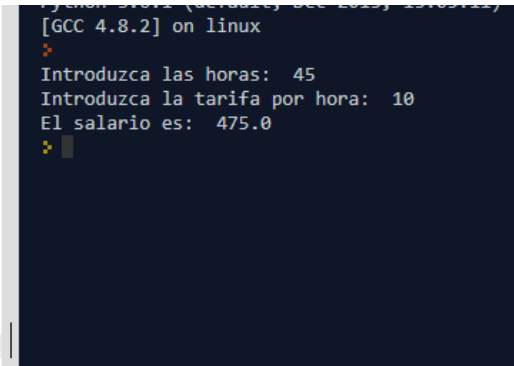
Ejercicio 4.6 Reescribe el programa de cálculo del salario, con tarifa-y-media para las horas extras, y crea una función llamada calculo_salario que reciba dos parámetros (horas y tarifa).

Introduzca Horas: 45

Introduzca Tarifa: 10

Salario: 475.0

```
1 def calculo_salario(horas, tarifa):  
2     if horas >= 40:  
3         horasext = horas - 40  
4         tarifaext = tarifa/2  
5         salario = (horas * tarifa)+(horasext*tarifaext)  
6         print("El salario es: ",salario)  
7     else:  
8         salario = horas * tarifa  
9         print("El salario es: ",salario)  
10  
11 horas = float (input("Introduzca las horas: "))  
12 tarifa = float(input("Introduzca la tarifa por hora: "))  
13 calculo_salario(horas,tarifa)  
14
```



Ejercicio 4.7 Reescribe el programa de calificaciones del capítulo anterior usando una función llamada calcula_calificacion, que reciba una puntuación como parámetro y devuelva una calificación como cadena.

Puntuación Calificación

> 0.9 Sobresaliente

> 0.8 Notable

> 0.7 Bien

> 0.6 Suficiente

<= 0.6 Insuficiente

Ejecución del programa:

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

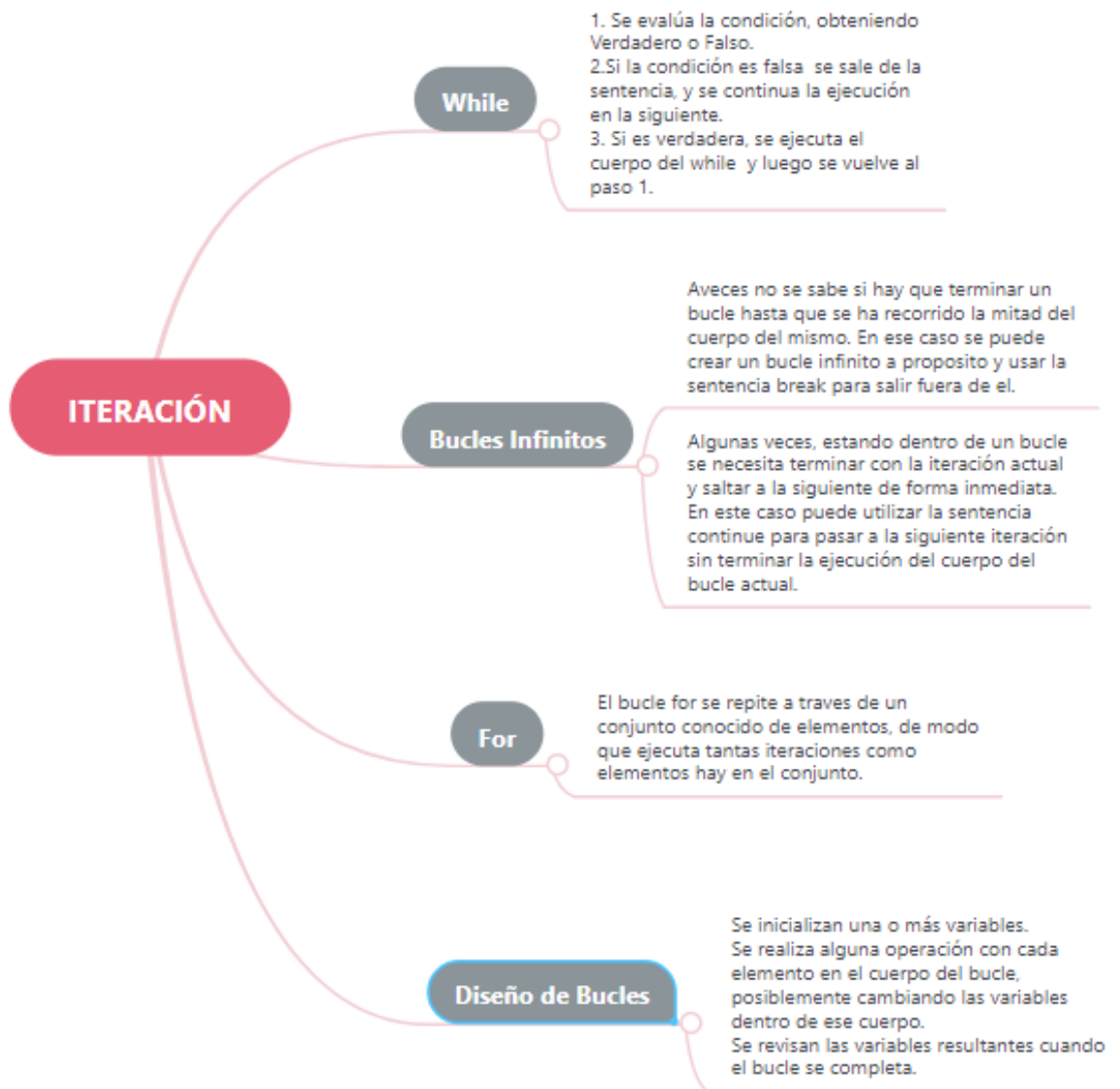
Insuficiente

Ejecuta el programa repetidamente para probar con varios valores de entrada diferentes.

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

```
[GCC 4.8.2] on linux
Ingrese la puntuacion entre 0 y 1: 0.95
Sobresaliente
Ingrese la puntuacion entre 0 y 1: perfecto
Puntuacion incorrecta
Ingrese la puntuacion entre 0 y 1: 10.0
Puntuacion incorrecta
Ingrese la puntuacion entre 0 y 1: 0.75
Bien
Ingrese la puntuacion entre 0 y 1: 0.5
Insuficiente
```

CAPITULO 5



Ejercicio 5.1 Escribe un programa que lea repetidamente números hasta que el usuario introduzca “fin”. Una vez se haya introducido “fin”, muestra por pantalla el total, la cantidad de números y la media de esos números. Si el usuario introduce cualquier otra cosa que no sea un número, detecta su fallo usando try y except, muestra un mensaje de error y pasa al número siguiente.

Introduzca un número: 4

Introduzca un número: 5

Introduzca un número: dato erróneo

Entrada invalida

Introduzca un número: 7

Introduzca un número: fin

16 3 5.333333333333333

```
1  numeros = []
2  suma = 0
3  op = input("Introducir numero?: ")
4  while op != 'fin':
5      try:
6          num = int(input("Introduce un numero: "))
7          suma = suma + num
8          numeros.append(num)
9          op = input("Seguir? ")
10         if op != 'si':
11             print("Entrada invalida")
12         else:
13             pass
14     except:
15         print("Entrada invalida")
16
17 promedio = suma/len(numeros)
18 print(suma, end='')
19 print(" ",len(numeros), end = " ")
20 print(" ",promedio)
21
```

```
[GCC 4.8.2] on linux
>
Introducir numero?: si
Introduce un numero: 4
Seguir? si
Introduce un numero: 5
Seguir? si
Introduce un numero: dato erroneo
Entrada invalida
Introduce un numero: 7
Seguir? fin
```

```
16 3 5.333333333333333
```

El programa muestra una ligera modificación comparándolo con la ejecución de ejemplo, ya que al implementarlo para no generar errores, se pregunta después de cada vez que se introduce un número si desea seguir o finalizarlo.

Ejercicio 5.2 Escribe otro programa que pida una lista de números como la anterior y al final muestre por pantalla el máximo y mínimo de los números, en vez de la media.

```
1 numeros = []
2 suma = 0
3 op = input("Introducir numero?: ")
4 while op != 'fin':
5     try:
6         num = int(input("Introduce un numero: "))
7         numeros.append(num)
8         op = input("Seguir? ")
9         if op != 'si':
10            print("Entrada invalida")
11        else:
12            pass
13    except:
14        print("Entrada invalida")
15
16 mayor = None
17 for i in numeros:
18     if mayor is None or i > mayor:
19         mayor = i
20 print('Mayor:', mayor)
21 menor = None
22 for i in numeros:
23     if menor is None or i < menor:
24         menor = i
25 print('Menor:', menor)
```

[GCC 4.8.2] on linux
Introducir numero?: si
Introduce un numero: 3
Seguir? si
Introduce un numero: 9
Seguir? si
Introduce un numero: -7
Seguir? si
Introduce un numero: 17
Seguir? si
Introduce un numero: 6
Seguir? si
Introduce un numero: 8
Seguir? fin

Mayor: 17
Menor: -7

- 2) Leer Capítulo III Análisis de la eficiencia de algoritmos de ordenamiento, 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. Este libro ya se proporcionó, y se pide hacer una lectura a conciencia y hacer un análisis profundo de los siguientes algoritmos:

Entre los algoritmos de ordenamiento *directos* tenemos:

- Ordenamiento por intercambio
- Ordenamiento por selección
- Ordenamiento por inserción
- Ordenamiento por burbuja

Entre los algoritmos de ordenamiento *indirectos* tenemos:

- Ordenamiento por mezcla (Mergesort)
- Ordenamiento rápido (Quicksort)
- Ordenamiento por Montículos (Heap sort)

También existen *otros* algoritmos de ordenamiento, como por ejemplo:

- Ordenamiento por Incrementos (Shell sort).
- Ordenamiento por Cubetas (Bin sort).
- Ordenamiento por Residuos (Radix).

“El nombre ordenamiento por burbuja se deriva del hecho de que los valores más pequeños en el arreglo flotan o suben hacia la parte inicial (primeras posiciones) del arreglo, mientras que los valores más grandes caen hacia la parte final (últimas posiciones) del arreglo. El algoritmo de ordenamiento por inserción ordena un arreglo de n elementos en orden ascendente, insertando directamente cada elemento de la lista en la posición adecuada y recorriendo hacia la derecha (de $[i]$ a $[i+1]$) los elementos restantes de la lista que son mayores al elemento insertado. Sea “a” el arreglo con los elementos a ordenar, el algoritmo comienza considerando una lista inicial compuesta por un solo elemento: $a[0]$.

A continuación se selecciona $a[1]$ y si éste es menor que $a[0]$, entonces $a[1]$ se inserta en la posición $[0]$ y este último se recorre una posición hacia la derecha, de lo contrario $a[1]$ se inserta en la posición $[1]$.

El proceso continúa seleccionando consecutivamente cada elemento de la sublista $a[i]$, $a[i+1]$, ..., $a[n-1]$, buscando la posición correcta para su inserción en la sublista $a[i-1]$, $a[i]$, ..., $a[0]$ y recorriendo hacia la derecha una posición todos los elementos mayores al elemento insertado.

El método de Selección

El algoritmo ejecuta varias pasadas (o iteraciones) y en cada una de ellas reemplaza el elemento del arreglo que está en la posición a substituir por el mínimo elemento encontrado. La primera posición a substituir es $a[0]$ y se analizará $a[0]$, $a[1]$, $a[2]$, ..., $a[n-1]$. La segunda posición a substituir es $a[1]$ y se analizará $a[1]$, $a[2]$, $a[3]$, ... $a[n-1]$, y así sucesivamente. En un arreglo de n elementos se ejecutan $n-1$ iteraciones.

Función hash o de dispersión es una función que mapea la clave de un elemento a un índice de un arreglo llamado tabla de dispersión. Cada elemento tiene una clave, y aplicando la función hash a esta clave se obtiene el índice i del arreglo o tabla de dispersión en donde se almacenará el elemento. Lo anterior se expresa como: $h(k) = i$

La función de Hash determina en donde se encuentra el apuntador a la lista ligada en donde se guardará o buscará el elemento

Una buena función de hash debe balancear la cantidad de elementos de los diferentes “pedazos” de lista.

A continuación se muestra una función hash para ordenar

Se parte de las siguientes premisas:

- Tenemos N elementos que se van a ordenar en B clases.
- Consideramos a los caracteres como enteros (su código ASCII).
- Suponemos que cada cadena tiene un máximo de 10 caracteres.

```
int hashFunction(char *nombre)
{
    int suma=0;
    for(int i=0;i<10;i++)
    {
        suma = suma + (int)nombre[i];
        return (suma%B);
    }
}
```

El código anterior es una función que regresa un número entre 0 y $B-1$, este número es el índice dentro de la tabla de dispersión, la cual es de tamaño B . El valor de hashFunction depende del valor ASCII de cada uno de los caracteres dentro de la cadena. “

- 3) Escribir un programa que le pida al usuario que ingrese una sucesión de números naturales (primero uno, luego otro, y así hasta que el usuario ingrese -1 como condición de salida). Al final, el programa debe imprimir cuántos números fueron ingresados, la suma total de los valores y el promedio.

```
1 def operaciones(n):
2     sum = 0
3     prom = 0
4     while n != -1:
5         num.append(n)
6         sum = sum + n
7         n = int(input("Ingresa un numero: "))
8     prom = sum/len(num)
9     print("La cantidad de numeros fue: ",len(num), ",y son: ",num)
10    print("La suma es: ",sum)
11    print("El promedio es: ", prom)
12
13    n = 0
14    num = []
15    n = int(input("Ingresa un numero: "))
16    if n == -1:
17        print("No hay numeros introducidos")
18    else:
19        operaciones(n)
```

```
[GCC 4.8.2] on linux
>
Ingresa un numero: 1
Ingresa un numero: 2
Ingresa un numero: 7
Ingresa un numero: 5
Ingresa un numero: 3
Ingresa un numero: 9
Ingresa un numero: -1
La cantidad de numeros fue: 6 ,y son: [1, 2, 7, 5, 3, 9]
La suma es: 27
El promedio es: 4.5
>
```

- 4) Escribir un programa que reciba una a una las notas del usuario, preguntando a cada paso si desea ingresar más notas, e imprimiendo el promedio correspondiente.

```
1 notas = []
2 sum = 0
3 prom = 0
4 op = int(input("1.Ingresar nota \nPresione cualquier numero para salir...\n"))
5
6 while op == 1:
7     n = int(input("Ingresa nota: "))
8     notas.append(n)
9     sum = sum + n
10    prom = sum/len(notas)
11    print("El promedio es: ",prom)
12    op = int(input("1.Ingresar nota \nPresione cualquier numero para salir...\n"))
13
```

```
1.Ingresar nota
Presione cualquier numero para salir...
1
Ingresa nota: 7
El promedio es: 7.0
1.Ingresar nota
Presione cualquier numero para salir...
1
Ingresa nota: 9
El promedio es: 8.0
1.Ingresar nota
Presione cualquier numero para salir...
1
Ingresa nota: 2
El promedio es: 6.0
1.Ingresar nota
Presione cualquier numero para salir...
1
Ingresa nota: 5
El promedio es: 5.75
1.Ingresar nota
Presione cualquier numero para salir...
1
Ingresa nota: 10
El promedio es: 6.6
1.Ingresar nota
Presione cualquier numero para salir...
1
Ingresa nota: 3
El promedio es: 6.0
1.Ingresar nota
Presione cualquier numero para salir...
```

5) Escribir funciones que dada una cadena de caracteres:

- a) Imprima los dos primeros caracteres.
- b) Imprima los tres últimos caracteres.
- c) Imprima dicha cadena cada dos caracteres. Ej.: `recta` debería imprimir `rca`
- d) Dicha cadena en sentido inverso. Ej.: `hola mundo!` debe imprimir `!odnum aloh`
- e) Imprima la cadena en un sentido y en sentido inverso. Ej: `reflejo` imprime `reflejoojelfer`.

```
1- def imprime2(cadena):
2-     print("\nLos dos primeros caracteres son: ",cadena[0],cadena[1])
3-
4- def ultimos3(cadena):
5-     print("\nLos ultimos tres caracteres son: ",cadena[-3],cadena[-2]
6-         ,cadena[-1])
7-
8- def cddos(cadena):
9-     print("\nImprimiendo cada dos caracteres: ")
10-    for i in range(0,len(cadena)):
11-        if i%2 == 0:
12-            print(cadena[i],end=' ')
13-
14- def invertir(cadena):
15-     print("\nLa cadena invertida es: ", cadena[::-1])
16-
17- def dos sentidos(cadena):
18-     print("\nLa cadena en dos sentidos: ",cadena+cadena[::-1])
19-
20-
21- cadena = (input("Ingresa una cadena: "))
22- imprime2(cadena)
23- ultimos3(cadena)
24- cddos(cadena)
25- invertir(cadena)
26- dos sentidos(cadena)
27-
```

```
[GCC 4.8.2] on linux
>
Ingresa una cadena: holamundo

Los dos primeros caracteres son:  h o

Los ultimos tres caracteres son:  n d o

Imprimiendo cada dos caracteres:
h l m n o
La cadena invertida es:  odnumaloh

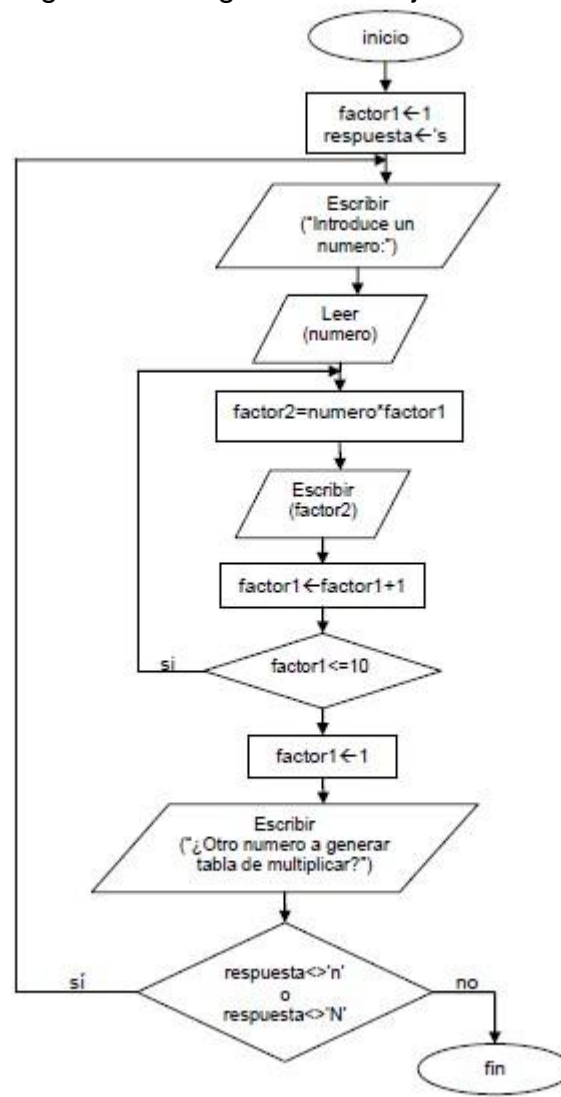
La cadena en dos sentidos:  holamundoodnumaloh
>
```

6) Escribir una función que reciba una cadena que contiene un largo número entero y devuelva una cadena con el número y las separaciones de miles. Por ejemplo, si recibe `1234567890`, debe devolver `1.234.567.890`.

```
1- def millares(cadena):
2-     str(cadena)
3-     cadena = cadena[::-1]
4-     ncadena = [cadena[i:i+3][::-1] for i in range(0,len(cadena),3) if
5-         (cadena[i:i+3])]
6-     ncadena = ncadena[::-1]
7-     cadena = str.join(".", ncadena)
8-     print("La cadena es: ",cadena)
9-
10- num = input("Ingresa una cadena de numeros: ")
11- millares(num)
```

```
[GCC 4.8.2] on linux
>
Ingresa una cadena de numeros: 1234567890
La cadena es:  1,234,567,890
>
```

7) Programar los siguientes Diagramas de flujo.



```

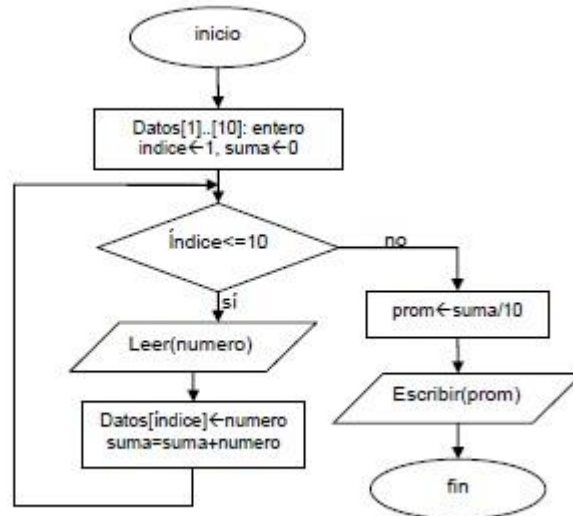
2 def main():
3     factor1 = 1
4     respuesta = 'si'
5     num = int(input("Ingresa un numero: "))
6     while factor1 <= 10:
7         factor2 = num * factor1
8         print(factor2)
9         factor1 = factor1 + 1
10    op = input("¿Otro numero a generar tabla de multiplicar?")
11    if op == respuesta:
12        main()
13
14    main()
15
16
17
18
19
20
21
22
23
24
25
26
27

```

```

>
Ingresa un numero: 1
1
2
3
4
5
6
7
8
9
10
¿Otro numero a generar tabla de multiplicar? si
Ingresa un numero: 2
2
4
6
8
10
12
14
16
18
20
¿Otro numero a generar tabla de multiplicar? no
>

```



```

1 indice = 1
2 suma = 0
3
4 while indice <= 10:
5     numero = int(input("Ingresa numero: "))
6     suma = suma + numero
7     indice = indice + 1
8
9 prom = suma/10
10 print("El promedio es: ",prom)
11
12
13
14
15

```

```

13:05:11)
[GCC 4.8.2] on linux
>
Ingresa numero: 1
Ingresa numero: 2
Ingresa numero: 3
Ingresa numero: 4
Ingresa numero: 5
Ingresa numero: 6
Ingresa numero: 7
Ingresa numero: 8
Ingresa numero: 9
Ingresa numero: 10
El promedio es: 5.5
>

```

8) Escribir funciones y programas que permitan encontrar:

- El máximo o mínimo de un polinomio de segundo grado (dados los coeficientes a, b y c), indicando si es un máximo o un mínimo.

```

1 def maxymin(a,b,c):
2     print("\nLa forma general es: {}x^2 {}x {}".format(a,b,c))
3     ad = 2*a
4     print("Primera derivada: {}x {}".format(ad,b))
5     print("Igualando a 0 y despejando x: 0 = {}x {}".format(ad,b))
6     x = -b/ad
7     print("El valor de x es: ",x)
8     print("Introduciendo x en la función original: ")
9     print("f({}) = {}({})^2 {}({}) {}".format(x,a,x,b,x,c))
10    temp = (a*(x**2)) + (b*x) + c
11    y = x/temp
12    print("El valor de y es: ",y)
13    if a>0:
14        print("\n----->Valor minimo<-----")
15        print("x = ",x, "\ny = ",y)
16    else:
17        print("\n----->Valor maximo<-----")
18        print("x = ",x, "\ny = ",y)
19
20    print("--Encontrando maximos y minimos de un polinomio--")
21    a=int(input("Ingrese el coeficiente a: "))
22    while a == 0:
23        a = int(input("Ingrese de nuevo a: "))
24    b=int(input("Ingrese el coeficiente b: "))
25    c=int(input("Ingrese el coeficiente c: "))
26    maxymin(a,b,c)
27

```

[GCC 4.8.2] on linux

--Encontrando maximos y minimos de un polinomio--

Ingrese el coeficiente a: 0

Ingrese de nuevo a: 3

Ingrese el coeficiente b: 6

Ingrese el coeficiente c: 8

La forma general es: 3x^2 6x 8

Primera derivada: 6x 6

Igualando a 0 y despejando x: 0 = 6x 6

El valor de x es: -1.0

Introduciendo x en la función original:

f(-1.0) = 3(-1.0)^2 6(-1.0) 8

El valor de y es: -0.2

----->Valor minimo<-----

x = -1.0

y = -0.2

- La intersección de dos rectas (dadas las pendientes y ordenada al origen de cada recta). Nota: validar que no sean dos rectas con la misma pendiente, antes de efectuar la operación.

```

1 def interseccion(m1,b1,m2,b2):
2     print("Recta 1: y = {}x {} \n Recta 2: y = {}x {}".format(m1,b1,m2,b2))
3     print("Igualamos ecuaciones: {}x {} = {}x {}".format(m1,b1,m2,b2))
4     temp1 = m1 - m2
5     temp2 = b2 - b1
6     x = temp2/temp1
7     print("El valor de x es: ",x)
8     print("Reemplazando el valor de x en alguna ecuación: y = {}({}) {}".format(m1,x,b1))
9     y = (m1*x) + b1
10    print("El valor de y es: ",y)
11    print("\n\n---->Coordenadas de intersección<---")
12    print("x = {} y = {}".format(x,y))
13
14    print("--Encontrando el punto de intersección de una recta (y = mx+b)--\n")
15    m1 = int(input("Ingrese la pendiente de la primer recta: "))
16    b1 = int(input("Ingrese la ordenada al origen de la primer recta: "))
17    m2 = int(input("Ingrese la pendiente de la segunda recta: "))
18    b2 = int(input("Ingrese la ordenada al origen de la segunda recta: "))
19    if m1 == m2:
20        print("Las rectas son paralelas, por lo tanto nunca se intersectan.")
21    else:
22        interseccion(m1,b1,m2,b2)
23

```

```
--Encontrando el punto de intersección de una recta (y = mx+b)--
```

```
Ingrese la pendiente de la primer recta: 1
Ingrese la ordenada al origen de la primer recta: 6
Ingrese la pendiente de la segunda recta: 1
Ingrese la ordenada al origen de la segunda recta: 8
Las rectas son paralelas, por lo tanto nunca se intersectan.
❖
```

```
--Encontrando el punto de intersección de una recta (y = mx+b)--
```

```
Ingrese la pendiente de la primer recta: -2
Ingrese la ordenada al origen de la primer recta: 12
Ingrese la pendiente de la segunda recta: 1
Ingrese la ordenada al origen de la segunda recta: 3
Recta 1: y = -2x 12
Recta 2: y = 1x 3
Igualamos ecuaciones: -2x 12 = 1x 3
El valor de x es: 3.0
Reemplazando el valor de x en alguna ecuación: y = -2(3.0) 12
El valor de y es: 6.0
```

```
--->Coordenadas de intersección<---
```

```
x = 3.0 y = 6.0
❖
```

9) Escribir funciones y programarlas que dada una cadena de caracteres:

- Devuelva solamente las letras consonantes. Por ejemplo, si recibe algoritmos o logaritmos debe devolver lgrtms.
- Devuelva solamente las letras vocales. Por ejemplo, si recibe sin consonantes debe devolver i ooae.

```
1 def vocales(cadena):
2     print("Las vocales contenidas son: ", end = '')
3     for i in range(0,len(cadena)):
4         if cadena[i]=='a' or cadena[i] == 'e' or cadena[i] == 'i' or cadena [i]
5             == 'o' or cadena[i] == 'u':
6             print(cadena[i], end = '')
7
8 def consonantes(cadena):
9     print("\nLas consonantes contenidas son: ", end = '')
10    for i in range(0,len(cadena)):
11        if cadena[i] == 'b' or cadena[i] == 'c' or cadena[i] == 'd' or cadena [i]
12            == 'f' or cadena[i] == 'g' or cadena[i] == 'h' or cadena[i] == 'j' or
13            cadena [i] == 'k' or cadena[i] == 'l' or cadena[i] == 'm' or cadena[i] ==
14            'n' or cadena[i] == 'ñ' or cadena [i] == 'p' or cadena[i] == 'q' or
15            cadena[i] == 'r' or cadena[i] == 's' or cadena [i] == 't' or cadena[i]
16            == 'v' or cadena[i] == 'w' or cadena[i] == 'x' or cadena[i] == 'y' or
17            cadena [i] == 'z':
18            print(cadena[i], end = '')
19
20 c = input("Ingresa cadena: ")
21 vocales(c)
22 consonantes(c)
```

```
[GCC 4.8.2] on linux
❖
Ingresa cadena: algoritmos
Las vocales contenidas son: aoio
Las consonantes contenidas son: lgrtms❖
```

Nota: Los programas están contenidos en el archivo Zip
grupo6_EDA_ProgramasTarea3