



SOLUCIONES EJERCICIOS

3ª PARTE TEMA 1: BUCLES

M^a Ángeles Fdez de Sevilla Vellón

3ªPARTE. TEMA1:

SOLUCIONES BUCLES

Número 1. Este programa calcula el producto de dos números enteros sin utilizar el operador de multiplicación (*).

```
# Archivo: ejercicio3_1.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Calcula el producto de dos enteros sin
# utilizar el operador multiplicación (*).

# Programa principal
x = int(input('Introduzca un número: '))
y = int(input('Introduzca otro número: '))
mul = 0

if (x == 0) or (y == 0):
    print()
else:
    for i in range (abs(y)):
        mul = mul + x
    if y < 0:
        mul = -mul

print(' la multiplicación es ', mul)
```

Número 2. Este programa calcula el valor de x elevado a y (x real, y entero).

```
# Archivo: ejercicio3_2.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Calcula el valor de x elevado a y.
#libreria
from math import *
# Programa principal
while True:
    base = float(input('Introduce base del número real: '))
    if base != 0:
        break
exponente = int(input('Introduce el exponente: '))
resultado = 1
for c in range (abs(exponente)):
    resultado = resultado * base
if exponente < 0:
    resultado = 1/resultado
```

```
print(base, ' elevado a ', exponente, ' = ', resultado)

# utilizando las funciones de logaritmo y exponenciación
resultado = e ** (exponente * log(abs(base)));
if (base < 0) and (exponente % 2 != 0):
    resultado = - resultado;
print(base, ' elevado a ', exponente, ' = ', resultado);
```

Número 3. Este programa pide una serie de números por pantalla -cantidad dada por el usuario- y calcula su media aritmética, el mayor y el menor.

NOTA: Para que funcione correctamente he inicializado la variable menor con un número muy alto.

```
# Archivo: ejercicio3_3.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Pide una serie de números por pantalla y
# calcula su media aritmética, el mayor y el menor.

# Llamada a librerías
from math import *
import sys

# Programa principal
while True:
    cuantos = int(input('Cuántos núm va a introducir?: '))
    if cuantos <= 0:
        print(';Tiene que ser una cantidad positiva!')
    else:
        break
suma = 0
mayor = -sys.maxsize
menor = sys.maxsize
for i in range (cuantos):
    numero = int(input('Introduce un numero: '))
    suma = suma + numero

    if numero < menor:
        menor = numero

    if numero > mayor:
        mayor = numero
media = suma / cuantos
print('La media de ', cuantos, ' números es: ', media)
print('El mayor es: ', mayor, 'y el menor es: ', menor)
```

Número 4. Este programa calcula los 20 primeros términos de la sucesión de Fibonacci y halla su suma.

```
# Archivo: ejercicio3_4.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Calcular los 20 primeros términos de la
# sucesión de Fibonacci.

# Constantes
Num_terminos = 20
Fib1, Fib2 = 1, 2

#Programa principal
fib_n = Fib1
print('Término N° 1 :', fib_n)
fib_nmas1 = Fib2
print('Término N° 2 :', fib_nmas1)
suma = fib_n + fib_nmas1

for i in range (3, num_terminos+1):
    fib_nmas2 = fib_nmas1 + fib_n
    fib_n = fib_nmas1
    fib_nmas1 = fib_nmas2
    print('Término N°', i,':', fib_nmas2)
    suma = suma + fib_nmas2

print('La suma de los ', Num_terminos,' primeros términos
es: ', suma)
```

Número 5. Este programa lee de pantalla 3 letras hasta leer la palabra fin (en mayúsculas o minúsculas).

```
# Archivo: ejercicio3_5.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Lee de pantalla 3 caracteres hasta leer la
palabra fin.

# Constantes
Carfin1 = 'F'
Carfin2 = 'I'
Carfin3 = 'N'

# Programa principal
while True:
    car1 = input('Introduzca primer carácter: ')
    car2 = input('Introduzca segundo carácter: ')
    car3 = input('Introduzca tercer carácter: ')
```

```
if (Carfin1 == car1.upper() and Carfin2 == car2.upper()
and Carfin3 == car3.upper()):
    break
```

Número 6. Este programa pide un entero positivo y calcula:

a) El número de dígitos necesario para escribirlo.

```
# Archivo: ejercicio3_6.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Lee un número entero positivo y calcula
# el número de dígitos necesario para escribirlo.
# Posteriormente a b calcula el número de veces que es
# divisible por 2.

#Constantes
Divisor = 2
Diez = 10
# Programa principal
while True:
    entero =int(input('Introduzca entero positivo: '))
    if entero < 0:
        print('; El entero tiene que ser positivo !')
    if entero > 0:
        break
num_digitos = 1
potencial0 = Diez
while potencial0 <= entero:
    num_digitos = num_digitos + 1
    potencial0 = potencial0 * 10
print('El entero ', entero, ' tiene ', num_digitos, '
dígito(s)')

#Apartado b
num_divisiones = 0
division = entero
while (division % Divisor == 0):
    division = division / Divisor
    num_divisiones = num_divisiones + 1

print('El entero ', entero, ' se puede dividir por ',
Divisor, ', ', num_divisiones, ' veces')
```

Número 7. Este programa lee una serie de enteros hasta que se introduzca -9999 y calcula el número total de enteros introducidos, cuántos son positivos y cuántos son negativos, utilizando el bucle while True.

```
# Archivo: ejercicio3_7.py
# Autor: XXXXX XXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Lee una serie de datos enteros hasta que se
# introduzca -9999 y calcula el número total de enteros
# introducidos, cuántos son positivos y cuantos negativos.
# Bucle repetir
# Constantes
Final = -9999

# Programa principal
total = 0
total_neg = 0
total_pos = 0
while True:
    entero = int(input('Introduzca un número entero ( -9999
para terminar) => '))
    if entero != Final:
        total = total + 1
        if (entero < 0):
            total_neg = total_neg + 1
        elif (entero>0):
            total_pos = total_pos +1
    else: # entero == Final
        break

print('Se han introducido ', total, ' números enteros')
print('Había ', total_pos,' positivos y ', total_neg, '
negativos')

# Archivo: ejercicio3_7B.py
# Autor: XXXXX XXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Lee una serie de datos enteros hasta que se
# introduzca -9999 y calcula el número total de enteros
# introducidos, cuántos son positivos y cuantos negativos.
# Bucle while más lógico con repetir
# Constantes
Final = -9999;
#Cuerpo principal
total = 0
total_neg = 0
total_pos = 0
entero = int(input('Introduzca un número entero ( -9999
para terminar) => '))
while entero!=Final:
    total=total+1;
```

```

if entero<0:
    total_neg = total_neg + 1
if entero>0:
    total_pos = total_pos + 1
entero = int(input('Introduzca un número entero (-9999
                    para terminar) => '))
print('Se han introducido ', total, ' números enteros')
print( total_pos, ' positivos y ', total_neg, ' negativos')

```

Número 8. Este programa pide un límite y calcula cuántos términos de la serie armónica son necesarios para que su suma supere dicho límite.

Serie armónica →
$$\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \dots$$

```

# Archivo: ejercicio3_8.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Pide un límite y calcula cuántos términos de
# la serie armónica son necesarios para que su suma supere
# dicho límite.

# Programa principal
limite = int(input('Introduzca el límite de la suma => '))
suma = 0
cuantos = 0

while True:
    cuantos = cuantos + 1
    suma = suma + 1 / cuantos
    if (suma > limite):
        break

print('El número de términos necesario es: ', cuantos)

```

Número 9. Este programa calcula la raíz cuadrada de una serie de números mediante el método de Newton, indicando el final de datos con un 0.

```

# Archivo: ejercicio3_9.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Calcula la raíz cuadrada de una serie de
# números mediante el método de Newton, indicando el final
# de datos con un 0.

# Llamada a la librería math entera
from math import *

```

```
# Constantes
Epsilon = 1E-6 # error cometido
Iteracion_max = 20 # número máximo de iteraciones
Final = 0 #indica el final de datos

# Programa principal
while True:

    while True:
        numero = float(input('Introduzca un número real
                               positivo ( 0 para terminar) => '))
        if numero >= 0:
            break

    if numero != Final:
        iteracion = 0
        if numero == 0:
            raiz = 0
        else:
            raiz = 1
            while True:

                raiz = (numero/raiz + raiz)/2
                iteracion = iteracion + 1
                if (abs(numero/sqrt(raiz) - 1) < Epsilon)
                or (iteracion == Iteracion_max):
                    break

        print('La raíz cuadrada de', numero, 'es:', raiz)
        if iteracion == Iteracion_max:
            print('(Nota: Se ha parado al llegar a ',
                  Iteracion_max, ' iteraciones)')

    if numero == Final:
        break
```

Número 10. Este programa lee un entero positivo (n).

- a) Genera una tabla con las n primeras potencias de 2
- b) Genera una tabla con las potencias de 2 menores o iguales que n.

```
# Archivo: ejercicio3_10.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Lee un número entero positivo y
# genera una tabla con las n primeras potencias de
# 2.

# Constantes
Base = 2

# Programa principal
while True:
    entero = int(input('Introduzca un entero
                        positivo => '))
    if entero <= 0:
        print('; El entero tiene que ser positivo
              !')
    if entero > 0:
        break

#Apartado A
print('* TABLA CON LAS', entero, 'PRIMERAS
      POTENCIAS DE', Base)
potencia = Base

for i in range(entero):
    print(Base, '^', i+1, '=', potencia)
    potencia = potencia * Base

#Apartado B

print('* TABLA CON LAS POTENCIAS DE', Base, 'MENORES
      O IGUALES QUE', entero)
potencia = Base
i = 1
while potencia <= entero:
    print(Base, '^', i, '=', potencia)
    potencia = potencia * Base
    i = i + 1
```

Número 11. Una pelota cae inicialmente desde una altura de dos metros. Si en cada bote la pelota sube hasta una altura igual al 95% de la que cayó en el bote anterior, hallar la distancia vertical recorrida en un total de 10 botes. ¿Cuántos botes deberá dar para que su altura sea menor de cinco centímetros?

```
# Archivo: ejercicio3_11.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa partiendo de la cauda
#de 2 metros de una pelota y del porcentaje de
#altura que alcanza en cada bote: halla la
#distancia vertical recorrida en un total de 10
#botes.
# Constantes
Porcent = 0.95
Inicial = 2
#Programa Principal
botes = 10
longitud = Inicial
altubote = longitud
for i in range(2, botes+1):

    altubote = altubote * Porcent
    longitud = longitud + 2*altubote
print (' en el bote numero',botes,' ha recorrido ')
print(longitud,' metros ')

#Posteriormente calcula el nº de botes que deberá
dar para que su altura sea menor de 5 cm
altubote = Inicial
i = 0
while True:
    altubote = altubote * Porcent
    i = i + 1
    if altubote < 0.05:
        break
print (' la altura despues de ',i,' botes es
',altubote)
```