

SOLUCIONES EJERCICIOS



TEMA 2: FUNCIONES Y PROCEDIMIENTOS

Número 1. Diseñar una función que calcule x^y .

```
# Archivo: ejercicio2_1.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa calcula la potencia de un número

# Definiciones de funciones

def eleva (x,y ):
    ''' (x,y)<- x^y Dados dos números eleva el primero al
    segundo'''
    resultado = 1
    for i in range (1, abs(y)+1) :
        resultado = resultado * x;
    if y<0:
        res = 1/resultado
    else:
        res = resultado
    return res

# Programa Principal

base = float(input ('dame base '))

if (base == 0):
    while True:
        exponente = int(input('la base es 0 dame exponente
distinto de 0'))
        if exponente != 0:
            break
else:
    exponente = int(input('Dame exponente'))
mipotencia= eleva (base, exponente)
print (' la potencia es ', mipotencia)
```

Número 2. Función booleana o lógica que pregunta al usuario si quiere continuar con el programa

```
#Archivo: ejercicio2_2.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
```

```
# Descripción: Programa para ver si queremos continuar

# Definiciones de funciones

def seguir():
    ''' () <- bool decide si salir de la función según carácter
        introducido por usuario'''
    while True:
        respuesta = str(input('¿QUIERES SALIR?: '))
        if (respuesta.upper()=='S') or (respuesta.upper() == 'N'):
            break
        r = (respuesta.upper() =='S')
    return r

# Programa Principal
if seguir():
    print('Quiero Continuar')
else:
    print('Salida de programa')
```

Número 3. programa que comprueba si un carácter dado es o no un dígito

```
#Archivo: ejercicio2_3.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Programa para comprobar si un carácter es un dígito

# Definiciones de funciones

def es_digito(car):
    '''car<- bool Dado un carácter devuelve verdad si es un
    dígito'''
    return (car>='0') and (car<='9')

# Programa Principal

c = input ('Escribe el carácter: ')
if es_digito(c):
    print('El carácter es un dígito')
else:
    print('El carácter no es un dígito')
```

Número 4. Función que recibe una base distinta de diez y un números en base decimal y escribe en pantalla el resultado de cambiar el número dado a la nueva base.

```

# Archivo: ejercicio2_4.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa recibe una base distinta de diez y un
# número en base decimal y escribe en pantalla el resultado de
# cambiar el número dado a la nueva base

# Definiciones de funciones
def cambio_base (num, base):
    ''' (num,base)<- numero Dado un número y una base devuelve el
    primer número cambiado de base'''
    Diez= 10;
    def eleva (x,y):
        ''' (x,y)<- x^y Dados dos números eleva el primero al
        segundo'''
        resultado = 1
        for i in range(1,abs(y)+1):
            resultado= resultado * x;
        if y<0:
            r = 1/resultado
        else:
            r = resultado
        return r
    numbase = 0
    i= 0
    while True:
        resto = num % base
        num= num // base
        numbase = numbase + resto * eleva(Diez,i)
        i= i+1;
        if (num == 0):
            break
    return numbase

#Programa Principal
dec = int(input('Dime un número'))
while True:
    base_inicial = int (input('base_inicial'))
    if (base_inicial >=2) and (base_inicial<=10):
        break

print ('el cambio de base es:',cambio_base (dec,base_inicial))

#Ejemplo cambio_base (13,2)= 1101{1+0*101+1*102+1*103

```

Número 5. Realizar un algoritmo MODULAR para obtener el máximo y mínimo de una serie de números enteros solicitados al usuario.

```
# Archivo: ejercicio2_5.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa obtiene el máximo y mínimo
#de una serie de números enteros solicitados al usuario.

# Definiciones de funciones

def seguir():
    ''' () <- bool decide si salir de la función según carácter
    introducido por usuario'''
    while True:
        respuesta = str(input('¿Desea continuar? '))
        if (respuesta.upper()=='S') or (respuesta.upper()=='N'):
            break
    return respuesta.upper()=='S'

def buscar_mayor_y_menor ( ):
    ''' () <- máximo, mínimoDevuelve el mayor y menor elemento de
    una serie de números'''
    dato = int(input('Introduzca dato '))
    maximo = dato
    minimo = dato
    while seguir():
        dato = int(input ('Inserte dato: '))
        if (dato> maximo):
            maximo = dato
        else:
            if (dato < minimo):
                minimo = dato
    return maximo, minimo

# Programa Principal

máximo, mínimo = buscar_mayor_y_menor ()
print(' El mayor elemento es ', máximo, 'el menor elemento es',
mínimo)
```

Número 6. Realizar un algoritmo para decidir si son o no primos una serie de números solicitados al usuario

```
# Archivo: ejercicio2_6.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa decide si son o no primos
# una serie de números enteros solicitados al usuario.

# Definiciones de funciones

def seguir():
    ''' () <- bool decide si salir de la función según dato
    introducido por usuario'''
```

```

while True:
    respuesta = str(input('¿Desea continuar? '))
    if (respuesta.upper()=='S') or (respuesta.upper()=='N'):
        break
return respuesta.upper()=='S'

def es_primo (n):
    '''entero<- booleano dado un número entero la función devuelve
    verdadero si n es primo y falso si no lo es '''
    if (n<= 1):
        primo = False
    else:
        aux = 2
        primo = True
        while (aux < n) and (primo):
            if (n % aux != 0):
                aux += 1
            else:
                primo = False;

        return primo
# Programa Principal
while True:
    n = int(input(' dame un numero '))
    if es_primo(n):
        print ('El numero es primo')
    else:
        print ('El numero no es primo')
    if not(seguir()):
        break

```

Número 7. Realizar un programa que lea un número arbitrario de enteros y cuente el número de valores positivos y números que se han introducido.

```

# Archivo: ejercicio2_7.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa lee un número arbitrario de enteros y
# cuenta el número de valores positivos que se han introducido.
# Definiciones de funciones
def seguir():
    ''' () <- bool decide si salir de la función según dato
    introducido por usuario'''
    while True:
        respuesta = str(input('¿Desea continuar? '))
        if (respuesta.upper()=='S') or (respuesta.upper()=='N'):
            break
    return respuesta.upper()=='S'

```

```
def cuenta_numeros_positivos (positivos, total):
    '''positivos, total <-positivos, total.Procedimiento en el que
    se introducen números mientras se desee, salida dos enteros:
    cantidad de números positivos y el total de números
    introducidos'''
    while True:
        numero = int(input('Introduce un numero integer : '))
        total =total + 1
        if numero > 0:
            positivos = positivos + 1
        if not(seguir()):
            break
    return positivos, total

# Programa principal

total_pos = 0
introducidos = 0
total_pos,introducidos = cuenta_numeros_positivos (total_pos,
introducidos)
print ('Se han Introducido ' , introducidos,' numeros enteros')
print(' de ellos ',total_pos, ' son Positivos ')
```

Número 8. Programa que comprueba si un carácter dado es una mayúscula del alfabeto inglés

```
# Archivo: ejercicio2_8.py
# Autor: XXXXX XXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa que compruebe si
# un carácter dado es una mayúscula del alfabeto inglés.
# Definición de funciones
def es_letra_mayuscula(car):
    '''caracter<- booleano dado un carácter esta función dice si
    pertenece o no a el alfabeto inglés'''
    return ((car >= 'A') and (car <= 'Z'))

# Programa principal

c = input(' Introduzca un carácter: ')

if es_letra_mayuscula(c):
    print(c, 'Es una letra mayúscula del alfabeto inglés')
else:
    print('no es una letra mayúscula del alfabeto inglés')
```

Número 9. Programa que comprueba si un carácter dado es una mayúscula del alfabeto español

```

# Archivo: ejercicio2_9.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Este programa que compruebe si
# un carácter dado es una mayúscula del alfabeto español.
# Definiciones de funciones
def es_letra_mayuscula(car):
    '''caracter<- booleano dado un carácter esta función dice si
    pertenece o no a el alfabeto español'''
    return ((car >= 'A') and (car <= 'Z'))or (car =='Ñ')

# Programa principal

c = input(' Introduzca un carácter: ')
if es_letra_mayuscula(c):
    print(c, 'Es una letra mayúscula del alfabeto español')
else:
    print('no es una letra mayúscula del alfabeto español')

```

Número 10. Este programa solicita una fecha y devuelve la fecha leída y la fecha 9 meses posterior escribiendo el mes en letra.

```

# Archivo: ejercicio2_10.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Solicita fecha y devuelve fecha leída y la
# fecha 9 meses mas tarde.

# Constantes
Num_meses = 9

# Definiciones de funciones
def lee_fecha():
    '''()<- día, mes , año: lectura de fecha comprobamos rangos'''
    while True:
        día = int(input('Día de nacimiento: '))
        if (día>=1) and (día <=31):
            break
    while True:
        mes = int(input('Mes de nacimiento: '))
        if (mes>=1) and (mes <=12 ):
            break
    while True:
        año = int(input('Año de nacimiento: '))
        if (año>=1900) and (año <=2100):
            break
    return día,mes,año

```

```
def escribe_fecha(dia,mes,año):
    ''' (dia, mes, año)<- cadena Lee un mes en número y lo
    devuelve n letra, el parámetro de entrada es el mes '''

    if mes == 1:
        mes_escrito = 'Enero'
    elif mes == 2:
        mes_escrito = 'Febrero'
    elif mes == 3:
        mes_escrito = 'Marzo'
    elif mes == 4:
        mes_escrito = 'Abril'
    elif mes == 5:
        mes_escrito = 'Mayo'
    elif mes == 6:
        mes_escrito = 'Junio'
    elif mes == 7:
        mes_escrito = 'Julio'
    elif mes == 8:
        mes_escrito = 'Agosto'
    elif mes == 9:
        mes_escrito = 'Septiembre'
    elif mes == 10:
        mes_escrito = 'Octubre'
    elif mes == 11:
        mes_escrito = 'Noviembre'
    else:
        mes_escrito = 'Diciembre'

    print(dia, ' DE ',mes_escrito,' DE ',año)

def fecha_mas_meses(dia,mes,año):
    ''' (dia, mes, año)<- Devuelve la fecha escrita 9 meses mas
    tarde de la primera leída '''

    if mes + Num_meses <= 12:
        escribe_fecha(dia,mes + Num_meses,año)
    else:
        escribe_fecha(dia , (mes + Num_meses)%12, año+(mes +
        Num_meses)//12)

# Programa principal

d,m,a = lee_fecha()
escribe_fecha(d,m,a)
fecha_mas_meses(d,m,a)
```

Número 11. Un cable de 1000 metros está tendido entre dos torres con una intermedia. Un funicular viaja por él a velocidad: $V=2.425+0.00175d^2$ si $d<30$ y $V=0.625+0.12d+0.00025d^2$

e.o.c, siendo d la distancia a la torre más cercana. Dada la distancia a la torre de la izquierda, calcular la velocidad e indicar distancia a la torre más cercana.

```
# Archivo: ejercicio2_11.py
# Autor: XXXXX XXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Calcular la velocidad de un funicular y la
# distancia a la torre más cercana.

# Llamada a la función sqrt de la librería math
from math import sqrt

# Definición de Constantes
Long_cable = 1000          # Longitud del cable
Mitad = Long_cable / 2     # Mitad de cable
Dist_limite = 30           # Distancia límite a una torre

# Definiciones de funciones

def cerca (p):
    ''' entero<- booleano: Indica si el funicular está a menos de
    30 metros de una torre, siendo 'p' la distancia a la torre de
    la izquierda. Función booleana, devuelve 1 si es verdad '''
    return (p <= Dist_limite) or ((p >= Mitad - Dist_limite) and (
    p <= Mitad + Dist_limite)) or (p>=(Long_cable - Dist_limite))

def dime_donde (pos):
    ''' posicion<- entero: Halla la distancia a la torre más
    cercana siendo 'pos' la distancia a la torre de la izquierda
    '''
    if pos > Mitad:
        pos = pos - Mitad

    if pos > (Mitad // 2):
        pos = Mitad - pos
    return pos

def velocidad (dis):
    ''' distancia<- real: Calcula la velocidad en función de la
    distancia al poste mas cercano '''
    if dis < Dist_limite:
        v = 2.425 + 0.00175*sqrt(dis)
    else:
        v = 0.625 + 0.12*dis - 0.00025*sqrt(dis)
    return v

# Programa principal

while True:
```

```
posicion = float(input('Introduzca la distancia al primer
                        poste: '))
if (posicion >= 0) and (posicion <= Long_cable):
    break

if cerca(posicion):
    print ('El funicular está a menos de ', Dist_limite, ' metros
           de una torre.')
else:
    print ('El funicular está a más de ', Dist_limite, ' metros de
           cualquier torre.')

donde = dime_donde(posicion)
print ('Cuando el funicular está a ', posicion, ' metros de la
       primera torre,')
print ('la más cercana está a ', donde, ' metros, ', 'y su
       velocidad es: ', velocidad(donde), ' m/s')
```

Número 12. Realizar un programa que dibuje con * un rectángulo dada su base y altura ó un triángulo de una altura dada por el usuario girado 90°

```
# Archivo: ejercicio2_12.py
# Autor: XXXXX XXXXXXXX (nombre y apellidos)
# Fecha: DD/MM/AAAA
# Descripción: Dibuja un rectángulo de una determinada base y
# altura ó un triángulo girado 90 grados de una altura determinada

# Definicion de funciones

def dibuja_rectangulo (x,y):
    ''' x,y <- dibuja con * un rectangulo de base x y altura y
    '''
    for i in range(y):
        for j in range(x):
            print('* ', end="")
        print()

def dibuja_triangulo (x):
    ''' Crea una forma de triángulo de altura x '''
    for i in range(1, x + 1):
        for j in range(i):
            print('* ', end="")
        print()

    for i in range(1, x):
        for j in range(x - i):
            print('* ', end="")
        print()
```

```
# Programa principal
```

```
forma = int(input('Introduce 1 para forma rectangular y 2 para  
forma triangular: '))
```

```
if forma == 1:  
    base = int(input('Base: '))  
    altura = int(input('Altura: '))  
    dibuja_rectangulo(base, altura)
```

```
elif forma == 2:  
    altura = int(input('Altura: '))  
    dibuja_triangulo (altura)
```

```
else:  
    print ('Error')
```