Instituto Superior Tecnológico
GUAYAQUIL

GUÍA GUÍA GENERAL DE LAS ACTIVIDADES DOCENTES

PRÁCTICAS

FUNDAMENTOS DE PROGRAMACIÓN

DEL INSTITUTO SUPERIOR TECNOLÓGICO

GUAYAQUIL

2023-2024

GUÍA GENERAL DE LAS ACTIVIDADES DOCENTES PRÁCTICAS.

Nombre de la asignatura: Fundamentos de programación

Carrera: Desarrollo de Software

Integrantes: Edgar Andrade, Elena Gallegos, Jimmy Totoy, Victor Padilla, Franklin Freire, Estefany Sánchez

Correo electrónico: eandrade@istg.edu.ec, egallegos@istg.edu.ec, jtotoy@istg.edu.ec, vpadilla@istg.edu.ec, ffreire@istg.edu.ec, esanchez@istg.edu.ec

Contenido

INTRODUCCIÓN

Esta Guía se fundamenta de los conocimientos básicos de programación, aplicados a ejercicios reales de las organizaciones y de los diferentes campos de conocimientos, donde se pueda automatizar procesos, con herramientas de programación utilizadas para el desarrollo en la actualidad.

En la Unidad 1, se abordan los conceptos fundamentales de programación enfocados en el lenguaje Python. Se exploran temas como el manejo de tipos de datos, variables y constantes, así como el trabajo con cadenas, lecturas por teclado, operadores lógicos, expresiones anidadas y operadores de asignación.

En la Unidad 2, se abordan conceptos y técnicas de programación que hacen uso de estructuras de control tales como if, while y for. Además, se exploran los tipos de datos estructurados como listas, tuplas, conjuntos y diccionarios. Se incluye la elaboración de menús de opciones que permiten la aplicación práctica de estas estructuras en diferentes contextos.

En la Unidad 3, se establece el entorno de desarrollo a utilizar y se profundiza en la definición de funciones, así como en el manejo del retorno y envío de valores dentro del contexto de la programación.

En la Unidad 4, se profundiza en los conceptos de argumentos y parámetros, así como en el uso de funciones integradas. Además, se exploran técnicas y diseños de algoritmos, donde se ponen en práctica diferentes métodos de resolución y se analiza su funcionamiento en profundidad.

Estas unidades contribuyen al desarrollo del perfil profesional de cada estudiante, ampliando sus conocimientos básicos en el ámbito de la programación.

OBJETIVO GENERAL

Planificar, escribir, desarrollar e implementar algoritmos de programación, a partir del análisis de requerimientos, teniendo en cuenta los criterios de calidad y ética profesional mediante el uso de Python como lenguaje de programación.

OBJETIVO ESPECÍFICO

1. Reconocer escenarios básicos y representarlos en pseudocódigos, desarrollando habilidades en el lenguaje de programación en Python de manera justificada y comprensible.
2. Interpretar algoritmos con estructuras de control, para consolidar las destrezas de desarrollo eficiente y de procedimientos complejos
3. Experimentar instrucciones y sentencias con entorno de desarrollo, utilizando funciones y retornos de valores, enfatizando la toma de decisiones en el proceso de programación.
4. Explorar casos prácticos con técnicas y diseño en algoritmos en Python para dar soluciones en un contexto tecnológico

UNIDAD 1: INTRODUCCIÓN A PYTHON

Objetivo

Reconocer escenarios básicos y representarlos en pseudocódigos, desarrollando habilidades en el lenguaje de programación en Python de manera justificada y comprensible.

Resultado de Aprendizaje

El estudiante comprenderá los conceptos básicos de pseudocodigo para aplicarlos en un lenguaje de programación de alto nivel.

1.1 Análisis Algorítmico

Actividad 1:

Crea un algoritmo en pseudocódigo que suma dos números:

Resolución:

Inicio

1. Solicitar al usuario que ingrese el primer número y guardarlo en la variable A
2. Solicitar al usuario que ingrese el segundo número y guardarlo en la variable B
3. Sumar A y B y guardar el resultado en la variable suma
4. Mostrar en pantalla el resultado de la suma

Fin

Actividad 2:

Escriba un algoritmos en pseudocódigo y flujogramas para la Suma de Dos Números

Resolución:

Pseudocódigo

Inicio

// Paso 1: Ingresar dos números

Leer num1

Leer num2

// Paso 2: Calcular la suma

suma = num1 + num2

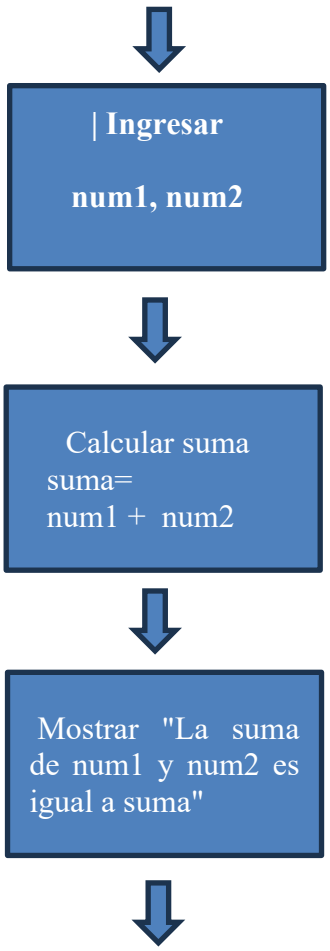
// Paso 3: Mostrar el resultado

Mostrar "La suma de", num1, "y", num2, "es igual a", suma

Fin

Flujograma

Inicio



Fin

1.2 Python como lenguaje de Programación

Actividad 1:

Investigación sobre la historia de la programación en Python y su evolución como lenguaje de programación.

Actividad 2:

Para familiarizarse con el entorno de desarrollo, práctica ejercicios básicos con python desde las diferentes herramientas (Pycharm, Visual Studio, QPython, Online Python Compiler), puedes tomar ejemplos de la web.

1.3 Tipos de datos

Actividad 1:

Práctica en el celular o en la PC para identificar tipos de datos interactuando con la aplicación

Resolución:

Cadena =" Tipos de datos Textos o cadenas"

Entero = 22

Decimal = 5.4

Actividad 2:

Escribe un programa que convierta una temperatura de grados Celsius a grados Fahrenheit. La fórmula de conversión es: Fahrenheit = Celsius * 9/5 + 32.

Resolución:

```
# Solicitar al usuario la temperatura en grados Celsius
celsius = 12.3

# Convertir Celsius a Fahrenheit
fahrenheit = celsius * 9/5 + 32

# Mostrar la temperatura en Fahrenheit
print("La temperatura en grados Fahrenheit es:", fahrenheit)
```

1.4 Lectura por teclado

Actividad 1:

Práctica de ejercicios en el celular o la PC con el uso de la **función input en Python**, para el ingreso de datos por teclado, se sugiere el ingreso de datos tipo registro de usuario (Ingreso de cédula, nombres, apellidos, direcciones, email, etc.)

Resolución:

```
# Solicitar al usuario ciertos datos
cedula = int(input("Ingresa su número de cédula: "))
cedula = int(input("Ingresa su número de cédula: "))
```

Actividad 2:

Escribe un programa que solicite al usuario su año de nacimiento y calcule su edad actual.

Resolución:

```
# Solicitar al usuario el año de nacimiento
year_of_birth = int(input("Ingresa tu año de nacimiento: "))

# Calcular la edad actual
current_year = 2024
age = current_year - year_of_birth

# Mostrar la edad
print("Tu edad es:", age)
```

1.5 Operadores Lógicos

Actividad 1:

Escribe un programa que solicite al usuario su edad y determine si es elegible para votar en las elecciones (mayor o igual a 18 años) y si es menor de edad.

Resolución:

```
# Solicitar la edad al usuario
edad = int(input("Ingrese su edad: "))

# Verificar si es mayor de edad y elegible para votar
```

```
if edad >= 18:
   print("¡Usted es elegible para votar!")
else:
   print("Usted es menor de edad y no puede votar aún.")
```

Actividad 2:

Escribe un programa que solicite al usuario las calificaciones de tres exámenes y determine si el estudiante aprobó todos los exámenes (calificación mayor o igual a 60 en cada uno).

Resolución:

```
# Solicitar las calificaciones de los exámenes
examen1 = float(input("Ingrese la calificación del primer examen: "))
examen2 = float(input("Ingrese la calificación del segundo examen: "))
examen3 = float(input("Ingrese la calificación del tercer examen: "))

# Verificar si todas las calificaciones son mayores o iguales a 60
if examen1 >= 60 and examen2 >= 60 and examen3 >= 60:
   print("¡Felicidades! Usted aprobó todos los exámenes.")
else:
   print("Lo siento, usted no aprobó todos los exámenes.")
```

1.6 Expresiones Anidadas

Actividad 1:

Escribe un programa que solicite al usuario su edad y determine en qué rango de edad se encuentra (niño, adolescente, adulto joven, adulto o adulto mayor).

Resolución:

```
# Solicitar la edad al usuario
edad = int(input("Ingrese su edad: "))

# Determinar el rango de edad
if edad < 0:
   print("Error: La edad ingresada es inválida.")
elif edad < 13:
   print("Usted es un niño.")
elif edad < 20:
   print("Usted es un adolescente.")
elif edad < 30:
```

```
    print("Usted es un adulto joven.")
elif edad < 60:
    print("Usted es un adulto.")
else:
    print("Usted es un adulto mayor.")
```

Actividad 2:

Escribe un programa que solicite al usuario un número y determine si es positivo, negativo o cero.

Resolución:

```
# Solicitar un número al usuario
numero = float(input("Ingrese un número: "))

# Determinar si el número es positivo, negativo o cero
if numero > 0:
    print("El número ingresado es positivo.")
elif numero < 0:
    print("El número ingresado es negativo.")
else:
    print("El número ingresado es cero.")
```

Estos ejercicios utilizan expresiones anidadas para realizar múltiples comparaciones y determinar acciones basadas en múltiples condiciones en Python. Puedes probar diferentes valores de entrada para ver cómo cambian los resultados.

1.7 Operadores de Asignación

Actividad 1:

Inicializa una variable contadora en 5 y utiliza un operador de asignación compuesta para incrementar su valor en 3.

Resolución:

```
# Inicializar la variable contadora en 5
contador = 5

# Utilizar el operador de asignación compuesta para incrementar en 3
contador += 3
```

```
# Mostrar el valor actual del contador
print("El valor del contador después de incrementarlo en 3 es:", contador)
```

Actividad 2:

Crea una cadena de texto y utiliza un operador compuesto para concatenar otra cadena

al final de la original.

Resolución:

```
# Crear una cadena de texto original
cadena_original = "Hola, "

# Concatenar otra cadena al final de la original utilizando el operador compuesto +=
cadena_original += "¿cómo estás?"

# Mostrar la cadena resultante
print("Cadena resultante:", cadena_original)
```

UNIDAD 2: ESTRUCTURAS DE CONTROL Y COLECICONES

Objetivo

Interpretar algoritmos con estructuras de control, para consolidar las destrezas de desarrollo eficiente y de procedimientos complejos.

Resultado de Aprendizaje

El estudiante será capaz de programar diferentes estructuras de control para ejecutar decisiones automatizadas en base a parámetros definidos

2.1 Estructuras de control

Actividad 1:

Crea una cadena y utiliza una sentencia if para imprimir un mensaje si la longitud de la cadena es mayor que 5.

Resolución:

```
# Crear una cadena de texto
cadena = "Python"

# Verificar si la longitud de la cadena es mayor que 5 usando una sentencia if
if len(cadena) > 5:
    print("La longitud de la cadena es mayor que 5.")
```

Actividad 2:

Dados dos números x e y, utiliza una estructura if-else para imprimir el número mayor.

Resolución:

```
# Definir dos números
x = 10
y = 20

# Utilizar una estructura if-else para imprimir el número mayor
```

```
if x > y:
   print("El número mayor es:", x)
else:
   print("El número mayor es:", y)
```

2.2 Sentencia IF

Actividad 1:

Escribe un programa que solicite al usuario ingresar un número y determine si es

positivo, negativo o cero.

Resolución:

```
# Solicitar al usuario ingresar un número
numero = float(input("Ingrese un número: "))

# Determinar si el número es positivo, negativo o cero
if numero > 0:
   print("El número es positivo.")
elif numero < 0:
   print("El número es negativo.")
else:
   print("El número es cero.")
```

Actividad 2:

Escribe un programa que solicite al usuario ingresar un número y determine si es par

o impar.

Resolución:

```
# Solicitar al usuario ingresar un número
numero = int(input("Ingrese un número entero: "))

# Verificar si el número es par o impar
if numero % 2 == 0:
   print("El número es par.")
else:
   print("El número es impar.")
```

2.3 Sentencia While

Actividad 1:

Utiliza un bucle while para imprimir los primeros 8 números naturales.

Resolución:

```
# Inicializar el contador
contador = 1

# Utilizar un bucle while para imprimir los primeros 8 números naturales
while contador <= 8:
    print(contador)
    contador += 1
```

Actividad 2:

Crea un programa que realice preguntas al usuario hasta que este responda

correctamente.

Resolución:

```
# Crear una cadena de texto original
# Definir la respuesta correcta
respuesta_correcta = "python"

# Inicializar una variable para almacenar la respuesta del usuario
respuesta_usuario = ""

# Utilizar un bucle while para hacer preguntas hasta que el usuario responda
correctamente
while respuesta_usuario != respuesta_correcta:
    # Solicitar al usuario que ingrese su respuesta
    respuesta_usuario = input("¿Cuál es un lenguaje de programación popular?
").lower()

    # Verificar si la respuesta del usuario es correcta
    if respuesta_usuario == respuesta_correcta:
        print("¡Correcto! Python es un lenguaje de programación popular.")
    else:
        print("Incorrecto. Intenta de nuevo.")

# Mensaje de despedida
```

print("¡Gracias por participar!")

2.4 Sentencia FOR

Actividad 1

Escribir un programa que pida al usuario una palabra y la muestre 10 veces por pantalla.

Resolución:

```
word = input("Introduce una palabra: ")
for i in range(10):
    print(word)
```

```
Introduce una palabra: HOLA
HOLA
HOLA
HOLA
HOLA
HOLA
HOLA
HOLA
HOLA
HOLA
HOLA
```

Actividad 2:

Escribir un programa que pida al usuario un número entero positivo y muestre por pantalla todos los números impares desde 1 hasta ese número separados por comas.

Resolución:

```
n = int(input("Introduce un número entero positivo: "))
for i in range(1, n+1, 2):
    print(i, end=", ")
```

```
Introduce un número entero positivo: 4
1, 3,
```

2.5 Listas

Actividad 1

Escribir un programa que almacene las asignaturas de un curso (por ejemplo Matemáticas, Física, Química, Historia y Lengua) en una lista y la muestre por pantalla.

Resolución:

```
subjects = ["Matemáticas", "Física", "Química", "Historia",
"Lengua"]
print(subjects)

['Matemáticas', 'Física', 'Química', 'Historia', 'Lengua']
```

Actividad 2:

Escribir un programa que almacene en una lista los números del 1 al 10 y los muestre por pantalla en orden inverso separados por comas.

Resolución:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for i in range(1, 11):
  print(numbers[-i], end=", ")
10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
```

2.6 Tuplas

Actividad 1:

Escribir un programa que almacene los vectores (1,2,3) y (-1,0,2) en dos tuplas y muestre por pantalla su producto escalar.

Resolución:

```
a = (1, 2, 3)
b = (-1, 0, 2)
product = 0
for i in range(len(a)):
    product += a[i]*b[i]
```

```
    print("El producto de los vectores" + str(a) + " y " + str(b) + "
     es " + str(product))
```

```
El producto de los vectores(1, 2, 3) y (-1, 0, 2) es 5
```

Actividad 2:

Crea una tupla con los meses del año, pide números al usuario, si el numero esta entre 1 y la longitud máxima de la tupla, muestra el contenido de esa posición sino muestra un mensaje de error. El programa termina cuando el usuario introduce un cero.

Resolución:

```
meses = ("Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio",
"Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre")

salir = False
while(not salir):

    numero = int(input("Dame un numero: "))

    if(numero==0):
        salir= True
    else:
        if(numero>=1 and numero<=len(meses)):
            print(meses[numero-1])
        else:
            print("Inserta un numero entre 1 y ",len(meses))
```

```
Dame un numero: 1
Enero
Dame un numero: 5
Mayo
Dame un numero: 0
```

2.7 Conjuntos

Actividad 1:

Crear un nuevo conjunto a partir del conjunto $A = \{1, 2, 3, 4, 5\}$ cuyos valores sean el cuadrado de los elementos del conjunto A.

Resolución:

```
A_ex = {1, 2, 3, 4, 5}

new_set = set ()
for e1 in A_ex:
  new_set.add (e1**2)

print (new_set)
```

```
{1, 4, 9, 16, 25}
```

Actividad 2:

El conjunto S y el conjunto T, sin usar el operador de intersección &, calcularemos la intersección de dichos conjuntos.

Resolución:

```
S = {1, 2, 3, 4}
T = {3, 4, 5, 6}

inter_set = set ()
for x in S:
  if x in T:
    inter_set.add (x)

inter_set
```

$\{3, 4\}$

2.8 Diccionarios

Actividad 1:

Escribir un programa que guarde en una variable el diccionario {'Euro':'€', 'Dollar':'$', 'Yen':'¥'}, pregunte al usuario por una divisa y muestre su símbolo o un mensaje de

aviso si la divisa no está en el diccionario.

Resolución:

```
monedas = {'Euro':'€', 'Dollar':'$', 'Yen':'¥'}
moneda = input("Introduce una divisa: ")
print(monedas.get(moneda.title(), "La divisa no está."))
```

```
Introduce una divisa: Dollar
$
```

Actividad 2:

Escribir un programa que pregunte una fecha en formato dd/mm/aaaa y muestre por pantalla la misma fecha en formato dd de de aaaa donde es el nombre del mes.

Resolución:

```
meses = {1:'enero', 2:'febrero', 3:'marzo', 4:'abril', 5:'mayo',
6:'junio', 7:'julio', 8:'agosto', 9:'septiembre', 10:'octubre',
11:'noviembre', 12:'diciembre'}
fecha = input('Introduce una fecha en formato dd/mm/aaaa: ')
fecha = fecha.split('/')
print(fecha[0], 'de', meses[int(fecha[1])], 'de', fecha[2])
```

```
Introduce una fecha en formato dd/mm/aaaa: 25/01/2024
25 de enero de 2024
```

UNIDAD 3: FUNCIONES EN PYTHON

Objetivo

Experimentar instrucciones y sentencias con entorno de desarrollo, utilizando funciones y retornos de valores, enfatizando la toma de decisiones en el proceso de programación.

Resultado de Aprendizaje

El estudiante podrá hacer uso de funciones y métodos en la construcción de un programa mediante el diseño de una solución utilizando instrucciones avanzadas.

3.1 Entorno de desarrollo

Actividad 1:

Desarrolla un programa en Python utilizando tu IDE que cuente la cantidad de palabras en un texto ingresado por el usuario.

Pseudocódigo:

- Solicitar al usuario que ingrese un texto.
- Dividir el texto en palabras usando el método split().
- Contar la cantidad de palabras obtenidas.
- Imprimir el número de palabras en el texto**.**

Actividad 2:

Crea un programa en Python utilizando tu IDE que genere una contraseña aleatoria de 8 caracteres que incluya letras mayúsculas, minúsculas, números y caracteres especiales.

Pseudocódigo:

- Definir una lista de caracteres que incluya letras mayúsculas, minúsculas, números y caracteres especiales.
- Usar la biblioteca random para seleccionar aleatoriamente caracteres de la lista y construir una contraseña de longitud 8.
- Imprimir la contraseña generada.
- Puedes implementar estos ejercicios en tu IDE de preferencia y ejecutarlos para verificar su funcionamiento.

3.2 Funciones

Actividad 1:

En la siguiente función determine el valor de retorno si **x** es igual a 5 y **y** es igual a 8:

```
def QuiénEsMayor(x,y):
    if x > y:
        return x
    else:
        return y
```

Resolución:

Al analizar el Código y evaluando la condición x>y, en el ejemplo al remplazar las variables 5>8 esta expresión es falsa, por lo tanto, la respuesta es 8

Actividad 2:

Escriba una función que reciba un número y retorne el 50% de este. El resultado que entrega la función será un valor lógico según corresponda.

Resolución:

```
def mitad_es_mayor_que_cinco(num):
    mitad = num / 2
    if mitad > 5:
        return True
    else:
        return False

# Ejemplo de uso
numero = 20
resultado = mitad_es_mayor_que_cinco(numero)
print("¿La mitad de", numero, "es mayor que 5?:", resultado)
```

Esta función mitad_es_mayor_que_cinco() toma un número como entrada, calcula el 50% de ese número y luego verifica si esa mitad es mayor que 5. Retorna True si la mitad es mayor que 5, de lo contrario retorna False. Puedes llamar a esta función con cualquier número como argumento y te dará el resultado deseado.

3.3 Definición de funciones

Actividad 1:

Declare una función llamada suma que reciba dos valores como parámetro

Resolución:

```
def suma(valor1, valor2):
        resultado = valor1 + valor2
        return resultado
```

```
# Ejemplo de uso

resultado_suma = suma(5, 3)

print(“La suma es:”, resultado_suma)
```

En este ejemplo, la función suma toma dos parámetros, valor1 y valor2, y devuelve la suma de estos dos valores. Luego, se muestra un ejemplo de cómo usar la función con los valores 5 y 3, y se imprime el 13. Puedes adaptar esta función según tus necesidades.

Actividad 2:

Declare una función llamada nombres completos que reciba dos textos como parámetros que serían los nombres y los apellidos.

Resolución:

```
def nombres_completos(nombre, apellidos):
        nombre_completo = nombre + “ “ + apellidos
        return nombre_completo
# Ejemplo de uso

nombre_usuario = input(“Ingresa tu nombre: “)
apellidos_usuario = input(“Ingresa tus apellidos: “)
nombre_completo_usuario = nombres_completos(nombre_usuario, apellidos_usuario)
print(“El nombre completo es:”, nombre_completo_usuario)
```

En este código, la función nombres_completos toma dos parámetros, nombre y apellidos, y devuelve la concatenación de ambos. Luego, se utiliza la función input para

solicitar al usuario que ingrese su nombre y apellidos, se llama a la función nombres_completos con esos valores y se imprime el resultado. Puedes ajustar el código según tus necesidades.

3.4 Retorno de valores

Actividad 1:

Realizar una función determine si un número es Par y retorne su respuesta:

Resolución:

```
def es_numero_par(numero):
    return numero % 2 == 0
# Ejemplo de uso

numero_usuario = int(input("Ingresa un número: "))
if es_numero_par(numero_usuario):
    print("El número ingresado es par")
else:
    print("El número ingresado es impar")
```

En este ejercicio, la función es_numero_par toma un número como parámetro y devuelve True si el número es par y False si es impar. Luego, se solicita al usuario ingresar un número, se llama a la función y se imprime el resultado.

Actividad 2:

Realizar una función determine el área de un círculo y retorne su respuesta. Se debe de utilizar como alternativa la librería math para el uso de la función integrada pi.

Resolución:

```
import math

def calcular_area_circulo(radio):
    area = math.pi * radio**2
    return area
# Ejemplo de uso

radio_usuario = float(input("Ingresa el radio del círculo: "))
area_resultante = calcular_area_circulo(radio_usuario)
print("El área del círculo es: ", area_resultante)
```

En este ejercicio, la función calcular_area_circulo toma el radio de un círculo como parámetro y devuelve el área calculada mediante la fórmula π

* radio^2.

3.5 Envió de valores

Actividad 1:

Realizar un ejercicio de función con envío de valores donde nos permita calcular el perímetro de un rectángulo.

Resolución:

```
def calcular_perimetro_rectangulo(lado1, lado2):
    perimetro = 2 * (lado1 + lado2)
    return perimetro
# Ejemplo de uso

lado1_rectangulo = float(input("Ingresa la longitud del primer lado del
rectángulo: "))
lado2_rectangulo = float(input("Ingresa la longitud del segundo lado del
```

```
rectángulo: “))
perimetro_resultante = calcular_perimetro_rectangulo(lado1_rectangulo,
lado2_rectangulo)
print(“El perímetro del rectángulo es:”, perimetro_resultante)
```

En este ejercicio, la función calcular_perimetro_rectangulo toma dos lados como parámetros y devuelve el perímetro de un rectángulo mediante la fórmula 2 * (lado1 + lado2).

Actividad 2:

Crea un programa en Python que solicite al usuario ingresar su nombre y edad. Luego, utiliza una función llamada generar_saludo para generar un saludo personalizado. La función deberá tomar el nombre y la edad como parámetros y devolverá un mensaje que incluya el saludo y la información proporcionada.

Resolución:

```
def generar_saludo(nombre, edad):
    saludo = “¡Hola, {}! Tienes {} años.”.format(nombre, edad)
    return saludo
# Ejemplo de uso

nombre_usuario = input(“Ingresa tu nombre: “)
edad_usuario = int(input(“Ingresa tu edad: “))
saludo_resultante = generar_saludo(nombre_usuario, edad_usuario)
print(saludo_resultante)
```

En este ejercicio, la función calcular_perimetro_rectangulo toma dos lados como parámetros y devuelve el perímetro de un rectángulo mediante la fórmula 2 * (lado1 + lado2).

UNIDAD 4: TÉCNICAS Y DISEÑO DE ALGORITMOS EN PYTHON

Objetivo

Explorar casos prácticos con técnicas y diseño en algoritmos en Python para dar soluciones en un contexto tecnológico

Resultado de Aprendizaje

En estudiante estará capacitado para analizar diferentes casos con programación avanzada

4.1 Argumentos y parámetros

Actividad 1:

Realice una función que reciba un número y una lista y agregue el número a la lista pasada por referencia. Realice un programa pida por teclado los valores a ingresar y pare cuando ingrese 0.

Resolución:

```
def agregar_numero_a_lista(numero, mi_lista):
    mi_lista.append(numero)

# Programa principal
numeros_ingresados = []
while True:
    valor = int(input("Ingresa un número (ingresa 0 para finalizar): "))
    if valor == 0:
    break
agregar_numero_a_lista(valor, numeros_ingresados)
print("Lista final:", numeros_ingresados)
```

En este programa:

1. La función agregar_numero_a_lista toma un número y una lista como
2. parámetros y utiliza el método append para agregar el número a la lista.
3. En el programa principal, se utiliza un bucle while True para solicitar números al usuario hasta que se ingresa 0.
4. Se llama a la función agregar_numero_a_lista para agregar cada número a la lista numeros_ingresados.
5. Cuando se ingresa 0, el bucle se rompe y se imprime la lista final.

Actividad 2:

Crea un programa en Python que utilice funciones para realizar peraciones matemáticas con dos números ingresados por el usuario. La función realizar_operacion debe recibir tres parámetros: numero1, numero2 y operacion. Dependiendo del valor de operacion, la función realizará la suma, resta, multiplicación o división de los dos números.

Resolución:

1. Define una función llamada realizar_operacion que tome tres parámetros: numero1 (número flotante), numero2 (número flotante) y operación (cadena de texto). La función debe devolver el resultado de la operación correspondiente.
    - Si operacion es “suma”, la función debe retornar la suma de numero1 y numero2.
    - Si operacion es “resta”, la función debe retornar la resta de numero1 y numero2.

- Si operacion es “multiplicacion”, la función debe retornar el producto de numero1 y numero2.
- Si operación es “division”, la función debe retornar la división de numero1 por numero2. Si numero2 es 0, el resultado debe ser un mensaje de error indicando que la división por cero no está permitida.
- Si operación no es ninguna de las anteriores, la función debe retornar un mensaje indicando que la operación no es válida.

2. En el programa principal, solicita al usuario que ingrese dos números (numero_1 y numero_2) y la operación que desea realizar (operacion_usuario).
3. Llama a la función realizar_operacion con los valores ingresados y almacena el resultado en una variable llamada resultado_operacion.
4. Imprime el resultado de la operación.

```
def realizar_operacion(numero1, numero2, operacion):
        if operacion == “suma”:
        resultado = numero1 + numero2
        elif operacion == “resta”:
        resultado = numero1 - numero2
        elif operacion == “multiplicacion”:
        resultado = numero1 * numero2
        elif operacion == “division”:
        if numero2 != 0:
        resultado = numero1 / numero2
        else:
        resultado = “Error: División por cero no permitida”
        else:
```

```
        resultado = “Operación no válida”
        return resultado
# Programa principal

numero_1 = float(input(“Ingresa el primer número: “))
numero_2 = float(input(“Ingresa el segundo número: “))
operacion_usuario = input(“Ingresa la operación a realizar (suma, resta,
multiplicacion, division): “)
resultado_operacion = realizar_operacion(numero_1, numero_2, operacion_usuario)
print(“Resultado de la operación:”, resultado_operacion)
```

En este ejercicio:

1. La función realizar_operacion toma tres parámetros: numero1, numero2 y operacion. Dependiendo del valor de operacion, realiza la operación matemática correspondiente.

2. En el programa principal, se solicita al usuario ingresar dos números y la operación que desea realizar.

3. Se llama a la función realizar_operacion con los valores ingresados y se imprime el resultado.

Este ejercicio ilustra cómo utilizar funciones con diferentes argumentos y parámetros para realizar operaciones matemáticas.

4.2 Funciones integradas

Actividad 1:

Escribe la instrucción que se debe utilizar para imprimir valores en la consola o en otro dispositivo de salida.

Resolución:

print(“Hola, mundo!”)

Actividad 2:

Genere una instrucción que devuelva la longitud (número de elementos) de un objeto, como una cadena, una lista, un diccionario.

Resolución:

```
cadena = “Python”
print(len(cadena)) # Output: 6
```

4.3 Casos prácticos (Algoritmos de ordenamiento)

Actividad 1:

Implementa el algoritmo de ordenamiento de burbuja para ordenar una lista de números de menor a mayor.

Resolución:

```
def bubble_sort(lista):
    n = len(lista)
    for i in range(n):
        for j in range(0, n-i-1):
            if lista[j] > lista[j+1]:
                lista[j], lista[j+1] = lista[j+1], lista[j]

# Ejemplo de uso
lista = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(lista)
print("Lista ordenada:", lista)
```

Actividad 2:

Implementa el algoritmo de ordenamiento por inserción para ordenar una lista de palabras alfabéticamente.

Resolución:

```
def insertion_sort(lista):
    for i in range(1, len(lista)):
        actual = lista[i]
        j = i-1
        while j >= 0 and actual < lista[j]:
            lista[j + 1] = lista[j]
            j -= 1
        lista[j + 1] = actual
# Ejemplo de uso
lista_palabras = ["perro", "gato", "elefante", "leon", "jirafa"]
insertion_sort(lista_palabras)
print("Lista ordenada:", lista_palabras)
```

Referencias Bibliográficas

Luján J, (2019), Aprendre a Programar con Python, AlfaOmega, ISBN: 9789587786026

Hernández M., Bquero L., (2021), Estructura de Datos - Fundamentos Prácticos, Ediciones de la U, ISBN: 9789587922707

Betancourt J, Polanco I, (2021), 115 Ejercicios Resueltos de Programación C++, Ediciones de la U, ISBN: 9789587922974

Castaño A. - Python fácil EDICIÓN 2DA., ISBN - 978-2-7460- LOS FUNDAMENTOS DEL LENGUAJE DE PYTHON 3

Rodríguez L. - Libro digital. Version 2.5 2016, Python Programación, Escuela Superior Politécnica del Litoral

Python. Notes for Professionals. Python®. Notes for Professionals, GoalKicker.com – Python® Notes for Professionals 185

Programación desde cero, 2018, Ejercicios resueltos de funciones en Python, http://patriciaemiguel.com/ejercicios/python/2019/03/10/ejercicios-funciones-python.html#:~:text=El%20siguiente%20programa%20deber%C3%ADa%20imprimir,%C2%BFQu%C3%A9%20hay%20que%20corregir%3F&text=Soluci%C3%B3n%3A%20Las%20funciones%20no%20utilizan,las%20variables%20globales%20x%2C%20y.

Condor E.,2020, Algoritmos resueltos con Python, Editorial EIDEC, https://www.editorialeidec.com/wp-content/uploads/2020/10/Algoritmos-resueltos-con-Python.pdf, DOI: https://doi.org/10.34893/6kbn-5a63

El libro de Python, 2022, Uso del If en Python, https://ellibrodepython.com/if-python#uso-del-if

Meza J. 2021, Uso de expresiones anidadas, https://www.programarya.com/Cursos/Python/Condicionales/Condicionales-anidados-elif

Báez L. Algoritmos y estructura de datos, 2020, http://www.luchonet.com.ar/aed/?page_id=209