



TECNOLOGICO JOSE MARIO MOLINA PASQUEL Y ENRIQUEZ UA ZAPOTLANEJO

MANUAL DE PROGRAMADOR



ESTRUCYURA DE DATOS





Contenido

Practica 1: Pide un numero entero, para después imprimir el valor elevado al cuadrado y también la raíz cuadrada del numero ingresado.	4
Practica 3:Este código lo que hace es que te pide ingresar 10 numeros los cuales se almacenan en un arreglo el cual después se imprime.	6
Programa 4: Aquí se hace algo parecido a lo que es el programa 3, pero ya validamos que si sean números con una simple asignación de caracteres.	7
Programa 5: Lo que se hace en este código es te pide letras o números, los números se almacenan en un arreglo y las letras en una lista, al final del código imprime lo que tenemos en el arreglo y en la lista también nos indica cuantos datos tenemos en cada uno.	8
Programa 6: Este código es similar al 5 hace exactamente lo mismo solo con validación.	9
Programa 7: Hacer un programa que lea nombre, edad y sexo de 5 personas, estos elementos tienen que estar dentro de una lista.	10
Programa 8: Hacer un programa que lea una cadena y que muestre en pantalla cuantos números tiene, cuantas mayúsculas, cuantas minúsculas y cuantos espacios	11
Programa 9: Hacer un programa que en una lista se introduzca cadenas de caracteres con las siguientes restricciones	12
Tarea1: Hacer un programa que lea 10 datos, si el dato es un número se almacenara en un arreglo si es un carácter o caracteres se meterá a una lista, cuando finalice el programa nos mostrará cuantos elementos números y cuantos caracteres hay en cada estructura.	14
Tarea2: Siguiendo los mismos planteamientos del programa9	15
Reparo1: Hacer un programa que lea nombre y precio de un producto el programa calculará el costo y precio de venta.	16
Reparo2: Fórmula general para ecuaciones cuadráticas	17
Reparo3: Hacer un programa que Lea un dato cual sea y que lo almacene en una lista, respetando su tipo de dato	18
ExamenPasado: Programa interactivo que solicita datos al usuario, realiza comparaciones y muestra los resultados en pantalla. Probablemente reproduce ejercicios de un examen anterior.	19
Programa1: Similar al anterior: pide datos al usuario, ejecuta condiciones y muestra mensajes según los valores ingresados.	20
Practica2: Usa una interfaz gráfica con Tkinter , permitiendo interactuar mediante ventanas y botones en lugar de la consola.	21
Programa3: También emplea Tkinter , posiblemente como segunda práctica con GUI, quizás mejorando el diseño o la funcionalidad del Programa2.	22
Programa4: Utiliza ciclos (for o while) para repetir acciones; puede manejar listas o crear menús con repeticiones controladas.	23
Programa5: Define funciones propias que devuelven resultados. Parece centrado en cálculos o validaciones específicas.	24





Programa6: Estructura modular con funciones reutilizables, posiblemente una continuación o mejora del Programa5...	26
Programa7: Similar a los dos anteriores: implementa funciones que organizan la lógica y los cálculos de manera estructurada.....	28
Repaso1: Incluye estructuras repetitivas para practicar bucles y condicionales; parece diseñado como ejercicio de repaso.	30
Repaso2: Otro archivo de repaso, con más ejemplos o variaciones del uso de ciclos y listas.....	31
Tarea1.1: Versión alternativa o mejorada de la tarea anterior, también con bucles o estructuras de control.....	33
Tarea1: Ejercicio práctico que emplea ciclos o menús para resolver tareas específicas; probablemente parte del segundo parcial.....	35
Validaciones: Archivo de apoyo que contiene funciones para validar datos (por ejemplo, verificar si una entrada es numérica o si cumple ciertas condiciones)	37
ValidacionRepasso: Versión de validación asociada al archivo Repaso2.py, usada para comprobar la entrada de datos del usuario	38
ValidarRepasso: Otro módulo de validación para los ejercicios de repaso; probablemente incluye funciones que evitan errores de entrada.	39
Tercera unidad, Programa1- Ordenamiento de números, y forma de eliminar en pilas y colas, el primero que entra es el ultimo que sale y el ultimo que entra es el primero que sale.....	40
Programa 2-Calculadora de RFC.....	43
Programa 3-Se pide lo que es nombre, numero de teléfono, correo. Agregándose a una lista y concatenándose para poderse mostrar.....	47
Programa 4-Tabla donde se introduce lo que es el nombre, edad, correo y la clave que se creó a base de los elementos ya mencionados y usando la librería random para agregar uno que otro valor random a la cadena.....	51
Unidad 4.....	55
Programa 1: El programa es una aplicación gráfica hecha en Tkinter que permite al usuario escribir dos números en una primera ventana. Al enviarlos, se abre una segunda ventana donde el usuario puede ver un mensaje y realizar la suma de los dos números ingresados, mostrando el resultado mediante un cuadro de diálogo. También incluye un botón para regresar a la ventana principal y volver a ingresar nuevos valores....	55
Programa 2: El programa es un sistema de login y administración de usuarios creado en Python con Tkinter y SQLite. Primero valida el acceso mediante un usuario y contraseña guardados en una base de datos ("Usuarios.db"). Si el inicio de sesión es correcto, abre una segunda ventana donde se muestra una tabla con todos los usuarios y se habilitan las opciones del menú según el rol del usuario (admin, supervisor o jefe de área). Dependiendo del rol, el usuario podrá agregar, modificar o eliminar cuentas. La tabla se actualiza dinámicamente y se evita eliminar al usuario que está actualmente en sesión.....	58
Programa 3: El programa es un sistema CRUD de productos hecho en Python con Tkinter y SQLite. Permite agregar, consultar, modificar y eliminar productos de una tienda, manejando dos tablas relacionadas (<i>productos</i> y <i>almacen</i>) mediante un INNER JOIN.	62
Tarea 1: El código realiza una consulta a la base de datos usando SQL para obtener información de productos y su respectivo inventario. Utiliza una sentencia INNER JOIN para unir la tabla <i>productos</i> con la tabla <i>almacen</i> mediante el código de producto, y devuelve datos como el ID, código, nombre del producto, precio, descripción del almacén y stock disponible.	66





Practica 1: Pide un numero entero, para después imprimir el valor elevado al cuadrado y también la raíz cuadrada del numero ingresado.

```

● ● ●
1 # int a = 0; # (ES) En C se declararía así, pero en Python no es necesario declarar el tipo | (EN) In C you would declare it like this, but in Python you don't need to declare the type
2
3 a = int(input('Escribe un numero: ')) # (ES) Pide al usuario un número entero | (EN) Asks the user for an integer
4 print(a**2) # (ES) Imprime el número elevado al cuadrado | (EN) Prints the number squared
5 print(a**(1/2)) # (ES) Imprime la raíz cuadrada del número | (EN) Prints the square root of the number
6
7
8 ##operadores aritméticos## # (ES) Lista de operadores aritméticos | (EN) List of arithmetic operators
9
10 # + # (ES) Suma | (EN) Addition
11 # - # (ES) Resta | (EN) Subtraction
12 # * # (ES) Multiplicación | (EN) Multiplication
13 # ** # (ES) Potencia | (EN) Exponentiation
14 # / # (ES) División con decimales | (EN) Division with decimals
15 # // # (ES) División entera (sin decimales) | (EN) Integer division (without decimals)
16 # % # (ES) Módulo, obtiene el residuo | (EN) Modulus, gets the remainder
17 # and, or # (ES) Operadores lógicos "y" / "o" | (EN) Logical operators "and" / "or"
18
19
20 ##Operadores diferenciales o lógicos## # (ES) Operadores de comparación y lógicos | (EN) Comparison and logical operators
21
22 # < # (ES) Menor que | (EN) Less than
23 # > # (ES) Mayor que | (EN) Greater than
24 # >= # (ES) Mayor o igual que | (EN) Greater than or equal to
25 # <= # (ES) Menor o igual que | (EN) Less than or equal to
26 # != # (ES) Diferente de | (EN) Not equal to
27 # not # (ES) Negación lógica | (EN) Logical negation
28 # == # (ES) Igual a | (EN) Equal to
29
30
31 # Para sacar un raíz cuadrada de un numero se debe de elevar el numero a la 1/2 # (ES) Explicación de cómo calcular raíz cuadrada | (EN) Explanation of how to calculate square root
32

```

Código programa 1

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python> & "C:/Users/Diego Plascencia/OneDrive/Escritorio/Practicas de Python/practicaspacial1/Programa1.py"
Escribe un numero: 45
2025
6.708203932499369
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python>

```

Código ejecución 1





Practica 2:

```

1 a = [10] # (ES) Se crea una lista con un solo elemento: 10 | (EN) A list is created with a single element: 10
2 b = [] # (ES) Se crea una lista vacía | (EN) An empty list is created
3 b.append(10000) # (ES) Se agrega el número 10000 a la lista b | (EN) Adds the number 10000 to list b
4 b = {'hola', 10, 10.05, False, 'm'} # (ES) Se reemplaza b con un conjunto (set) que contiene varios elementos | (EN) b is replaced with a set containing several elements
5
6 print(a[0]) # (ES) Imprime el primer elemento de la lista a, que es 10 | (EN) Prints the first element of list a, which is 10
7
8 if len(a) > len(b): # (ES) Compara el tamaño de la lista a con el del conjunto b | (EN) Compares the size of list a with that of set b
9     print('A es mayor') # (ES) Se imprime si la lista a tiene más elementos | (EN) Prints if list a has more elements
10 else:
11     print('B es mayor') # (ES) Se imprime si el conjunto b tiene más o igual cantidad de elementos | (EN) Prints if set b has more or the same number of elements
12
13 for i in a: # (ES) Recorre cada elemento de la lista a | (EN) Iterates over each element in list a
14     print(a) # (ES) Imprime toda la lista a en cada iteración (a = [10]) | (EN) Prints the entire list a in each iteration (a = [10])
15

```

Código programa 2

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python> &
cia/OneDrive/Escritorio/Practicas de Python/practicaspacial1/Programa2.
10
B es mayor
[10]
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python>

```

Código ejecución 2





Practica 3:Este código lo que hace es que te pide ingresar 10 numeros los cuales se almacenan en un arreglo el cual después se imprime.

```

1 # Hacer un programa que lea 10 numeros # (ES) El programa debe leer 10 números | (EN) The program must read 10 numbers
2 # Y los almacene en un arreglo # (ES) Guardarlos en un arreglo (lista en Python) | (EN) Store them in an array (list in Python)
3
4 a = [0,0,0,0,0,0,0,0,0] # (ES) Se crea una lista con 10 elementos inicializados en 0 | (EN) A list with 10 elements initialized at 0 is created
5
6 #Para formatear el dato pones un f antes de cadena y al final orchetas
7 # (ES) Para usar f-strings se coloca una f antes de la cadena y llaves {} dentro | (EN) To use f-strings put an f before the string and braces {} inside
8
9 """for i in range[0:10]:
10     a[i] = int(input(f'Escribe un numero{i+1}'))"""\# (ES) Ejemplo comentado con errores de sintaxis | (EN) Example commented with syntax errors
11
12 for i in range(0,10): # (ES) Recorre los indices de 0 a 9 (10 posiciones) | (EN) Loops through indexes from 0 to 9 (10 positions)
13     a[i] = int(input('Escribe un numero: \n')) # (ES) Pide un número al usuario y lo guarda en la posición i | (EN) Asks the user for a number and stores it in position i
14
15 for i in a: # (ES) Recorre cada número dentro de la lista a | (EN) Iterates through each number in list a
16     print(i) # (ES) Imprime cada número ingresado | (EN) Prints each entered number
17

```

Código programa 3

```

Escribe un numero:
45
Escribe un numero:
78
45
78
45
45
45
78
45
12
45
78
PS C:\Users\Diego Plascencia\OneDri

```

Código ejecución 3





Programa 4: Aquí se hace algo parecido a lo que es el programa 3, pero ya validamos que si sean números con una simple asignación de caracteres.

```

1 # Hacer un programa que lea 10 numeros    # (ES) Programa que lee 10 números | (EN) Program that reads 10 numbers
2 # Y los almacene en una lista           # (ES) Los guarda en una lista | (EN) Stores them in a list
3
4 a = [] # (ES) Se inicializa la lista vacía | (EN) The empty list is initialized
5 suma = 0 # (ES) Variable para acumular la suma de los números | (EN) Variable to accumulate the sum of numbers
6 n = 0 # (ES) Contador para controlar cuántos números se han introducido | (EN) Counter to control how many numbers have been entered
7 numeros = "0,1,2,3,4,5,6,7,8,9"; # (ES) Cadena usada para validar que se ingresen solo dígitos | (EN) String used to validate only digits are entered
8
9 while(n <= 9): # (ES) Bucle que se repite hasta que se ingresen 10 números | (EN) Loop repeats until 10 numbers are entered
10    b = input('Escribe un numero: ') # (ES) Pide al usuario un número como texto | (EN) Asks the user for a number as text
11    x = 0 # (ES) Contador para validar si todos los caracteres son números | (EN) Counter to check if all characters are digits
12    for i in b: # (ES) Recorre cada carácter del valor ingresado | (EN) Iterates through each character of the entered value
13        if(ord(i) >= 48 and ord(i) <= 57): # (ES) Opción: validar usando el código ASCII de los números | (EN) Option: validate using ASCII code of digits
14            if i in numeros: # (ES) Valida si el carácter está en la cadena de números | (EN) Validates if the character is in the string of digits
15                x += 1 # (ES) Si es número, incrementa el contador | (EN) If it's a digit, increment the counter
16            if len(b) == x: # (ES) Si la cantidad de dígitos coincide con la longitud del texto ingresado, es válido | (EN) If digit count matches length of input, it's valid
17                a.append(int(b)) # (ES) Convierte el texto a entero y lo guarda en la lista | (EN) Converts text to integer and stores it in the list
18                n += 1 # (ES) Aumenta el contador porque se ingresó un número válido | (EN) Increases counter because a valid number was entered
19        else:
20            print('El valor no es un número') # (ES) Mensaje si lo ingresado no es válido | (EN) Message if the input is not valid
21
22 for i in a: # (ES) Recorre la lista con los números ingresados | (EN) Iterates through the list with the entered numbers
23     print(i) # (ES) Imprime cada número | (EN) Prints each number
24     suma += i # (ES) Suma cada número a la variable suma | (EN) Adds each number to the sum variable
25
26 print(f'La suma es: {suma}') # (ES) Imprime el total de la suma | (EN) Prints the total sum
27

```

Código programa 4

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python> & "C:/cia/OneDrive/Escritorio/Prácticas de Python/prácticasparciales1/Programa4.py"
Escibe un numero: 45
Escibe un numero: 45
Escibe un numero: 45
Escibe un numero: 45
Escibe un numero: s
El valor no es un numero
Escibe un numero: 78
Escibe un numero: x
El valor no es un numero
Escibe un numero: 7
45
45
45
45
78
7
7
7
7
7
La suma es: 293
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python>

```

Código ejecución 4





Programa 5: Lo que se hace en este código es te pide letras o números, los números se almacenan en un arreglo y las letras en una lista, al final del código imprime lo que tenemos en el arreglo y en la lista también nos indica cuantos datos tenemos en cada uno.

```

1 arr = [0,0,0,0,0,0,0,0,0,0] # (ES) Arreglo con 10 ceros para guardar números | (EN) Array with 10 zeros to store numbers
2 lis = [] # (ES) Lista vacía para guardar letras/cadenas | (EN) Empty list to store letters/strings
3 c = 0 # (ES) Contador de entradas totales (hasta 10) | (EN) Counter of total inputs (up to 10)
4 c2 = 0 # (ES) Contador de números válidos en arr | (EN) Counter of valid numbers in arr
5
6 while(True): # (ES) Bucle infinito que se detendrá con break | (EN) Infinite loop that stops with break
7     a = input('Escribe un dato o valor \n') # (ES) Pide un dato al usuario | (EN) Asks the user for a value
8     c += 1 # (ES) Incrementa el contador de entradas | (EN) Increases the input counter
9     if a.isdigit(): # (ES) Valida si lo ingresado son solo dígitos (número entero positivo) | (EN) Checks if input is only digits (positive integer)
10        arr[c-1] = int(a) # (ES) Convierte el valor a entero y lo guarda en arr | (EN) Converts input to integer and stores it in arr
11    elif a.isalpha(): # (ES) Valida si lo ingresado son solo letras | (EN) Checks if input is only letters
12        lis.append(a) # (ES) Agrega el texto a la lista lis | (EN) Appends the text to list lis
13    if c >= 10: # (ES) Si ya se ingresaron 10 datos, termina el bucle | (EN) If 10 values entered, stop the loop
14        break
15
16 for i in arr: # (ES) Recorre el arreglo arr | (EN) Iterates through array arr
17     if i != 0: # (ES) Cuenta solo los números distintos de 0 (los ingresados) | (EN) Counts only numbers different from 0 (the entered ones)
18         c2 += 1 # (ES) Incrementa el contador de números válidos | (EN) Increases valid number counter
19
20 print(f'El arreglo tiene {c2}') # (ES) Imprime cuántos números se guardaron | (EN) Prints how many numbers were stored
21 print(f'La lista tiene {len(lis)}') # (ES) Imprime cuántas cadenas se guardaron | (EN) Prints how many strings were stored
22 print(arr) # (ES) Imprime todo el arreglo de números | (EN) Prints the whole number array
23 print(lis) # (ES) Imprime toda la lista de letras/cadenas | (EN) Prints the whole string list
24

```

Código programa 5

```

Escribe un dato o valor
asd
Escribe un dato o valor
ASF
Escribe un dato o valor
ASF
Escribe un dato o valor
ASF
Escribe un dato o valor
sa
Escribe un dato o valor
asd
Escribe un dato o valor
asd
Escribe un dato o valor
457
Escribe un dato o valor
457
Escribe un dato o valor
457
El arreglo tiene 3
La lista tiene 7
[0, 0, 0, 0, 0, 0, 0, 457, 457, 457]
['asd', 'ASF', 'ASF', 'sa', 'asd', 'asd']
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python>

```

Código ejecución 5





Programa 6: Este código es similar al 5 hace exactamente lo mismo solo con validación.

```

● ● ●

1 def hola(): # (ES) Función principal que pide los datos | (EN) Main function that asks for the inputs
2     c = 0 # (ES) Contador de entradas | (EN) Input counter
3     while(True): # (ES) Bucle infinito controlado por break | (EN) Infinite loop controlled by break
4         a = input('Escribe un dato o valor \n') # (ES) Pide al usuario un valor | (EN) Asks the user for a value
5         c += 1 # (ES) Incrementa el contador | (EN) Increments the counter
6         if a.isdigit(): # (ES) Si lo ingresado son solo dígitos (número positivo) | (EN) If input is only digits (positive number)
7             arr[c-1] = int(a) # (ES) Convierte el dato a entero y lo guarda en el arreglo en la posición correspondiente | (EN) Converts input to integer and stores it in array at the right position
8             elif a.isalpha(): # (ES) Si lo ingresado son solo letras | (EN) If input is only letters
9                 lis.append(a) # (ES) Se agrega a la lista de cadenas | (EN) Appends it to the string list
10            if c >= 10: # (ES) Si ya se ingresaron 10 valores, se rompe el bucle | (EN) If 10 values are entered, break the loop
11                break
12        resultados() # (ES) Llama a la función que muestra los resultados | (EN) Calls the function that shows results
13
14 def resultados(): # (ES) Función que imprime los resultados | (EN) Function that prints results
15     c2 = 0 # (ES) Contador de números válidos en arr | (EN) Counter of valid numbers in arr
16     for i in arr: # (ES) Recorre el arreglo | (EN) Iterates through the array
17         if i != 0: # (ES) Si el valor es distinto de 0, significa que se ingresó un número | (EN) If value is not 0, it means a number was entered
18             c2 += 1 # (ES) Incrementa el contador de números | (EN) Increments number counter
19     print(f'El arreglo tiene {c2}') # (ES) Imprime la cantidad de números guardados | (EN) Prints how many numbers were stored
20     print(f'La lista tiene {len(lis)}') # (ES) Imprime la cantidad de cadenas guardadas | (EN) Prints how many strings were stored
21     print(arr) # (ES) Imprime el arreglo completo | (EN) Prints the whole array
22     print(lis) # (ES) Imprime la lista completa | (EN) Prints the whole list
23
24 arr = [0,0,0,0,0,0,0,0] # (ES) Arreglo de 10 posiciones inicializado en 0 | (EN) Array of 10 positions initialized at 0
25 lis = [] # (ES) Lista vacía para guardar cadenas | (EN) Empty list to store strings
26
27 if __name__ == "__main__": # (ES) Punto de entrada del programa | (EN) Entry point of the program
28     hola() # (ES) Llama a la función principal | (EN) Calls the main function

```

Código programa 6

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python> & "C:/Users/cia/OneDrive/Escritorio/Prácticas de Python/prácticasparcial1/Programa6.py"
Escribe un dato o valor
d
Escribe un dato o valor
78
Escribe un dato o valor
78
Escribe un dato o valor
7s
Escribe un dato o valor
78
Escribe un dato o valor
Escribe un dato o valor
4
El arreglo tiene 4
La lista tiene 4
[0, 0, 0, 0, 78, 78, 0, 78, 0, 4]
['d', 'd', 'd', 'd']
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python>

```

Código ejecución 6





Programa 7: Hacer un programa que lea nombre, edad y sexo de 5 personas, estos elementos tienen que estar dentro de una lista

```

1 def pedir(): # (ES) Función para pedir los datos | (EN) Function to request data
2     d = 0 # (ES) Contador de personas registradas | (EN) Counter for registered people
3     while(True): # (ES) Bucle infinito que termina con break | (EN) Infinite loop that ends with break
4         nombre = input('Cual es tu nombre: ') # (ES) Pide el nombre | (EN) Asks for the name
5         edad = input('Cual es tu edad: ') # (ES) Pide la edad | (EN) Asks for the age
6         sexo = input('Cual es tu genero M o H: ') # (ES) Pide el sexo (M = mujer, H = hombre) | (EN) Asks for gender (M = female, H = male)
7         lista.append("Nombre: "+nombre+", Edad: "+edad+", Sexo: "+sexo)
8         # (ES) Une los datos en un texto y los agrega a la lista | (EN) Joins the data in a string and appends it to the list
9         d += 1 # (ES) Incrementa el contador de personas | (EN) Increases the people counter
10        if d >= 5: # (ES) Cuando ya se registraron 5 personas, termina el bucle | (EN) When 5 people have been registered, stops the loop
11            break
12    resultado() # (ES) Llama a la función que muestra los resultados | (EN) Calls the function that shows results
13
14 def resultado(): # (ES) Función que imprime la lista de personas | (EN) Function that prints the people list
15     print(lista) # (ES) Muestra toda la lista | (EN) Displays the whole list
16
17 lista = [] # (ES) Lista vacía para guardar la información | (EN) Empty list to store the information
18
19 if __name__ == "__main__": # (ES) Punto de entrada del programa | (EN) Entry point of the program
20     pedir() # (ES) Llama a la función pedir() | (EN) Calls the ped

```

Código programa 7

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python> & "C:/Users/Diego Plascencia/Applicación/OneDrive/Escritorio/Practicas de Python/practicaspacial1/Programa7.py"
Cual es tu nombre: Diego
Cual es tu edad: 78
Cual es tu genero M o H: H
Cual es tu nombre: Diego
Cual es tu edad: 74
Cual es tu genero M o H: h
Cual es tu nombre: ew
Cual es tu edad: ddew
Cual es tu genero M o H: 45
Cual es tu nombre: m
Cual es tu edad: 78
Cual es tu genero M o H: dwe
Cual es tu nombre: dwe
Cual es tu edad: wed
Cual es tu genero M o H: wed
['Nombre: Diego, Edad: 78, Sexo: H', 'Nombre: Diego, Edad: 74, Sexo: h', 'Nombre: ew, Edad: ddew, Sexo: 45', 'Nombre: m, Edad: 78, Sexo: dwe', 'Nombre: wed, Edad: wed, Sexo: wed']
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python>

```

Código ejecución 7





Programa 8: Hacer un programa que lea una cadena y que muestre en pantalla cuantos números tiene, cuantas mayúsculas, cuantas minúsculas y cuantos espacios

```

● ● ●
1 def inicio(): # (ES) Función principal | (EN) Main function
2     n = 0 # (ES) Contador de números | (EN) Counter for numbers
3     e = 0 # (ES) Contador de espacios | (EN) Counter for spaces
4     min = 0 # (ES) Contador de minúsculas | (EN) Counter for lowercase
5     m = 0 # (ES) Contador de mayúsculas | (EN) Counter for uppercase
6     numeros = "0123456789" # (ES) Cadena usada para validar números | (EN) String used to validate numbers
7     cadena = input('Escribe una cadena: ') # (ES) Pide una cadena al usuario | (EN) Asks the user for a string
8     for i in cadena: # (ES) Recorre cada carácter de la cadena | (EN) Iterates through each character of the string
9         if i in numeros: # (ES) Si el carácter está en "0123456789" es número | (EN) If character is in "0123456789", it's a number
10            n += 1
11        if i == ' ': # (ES) Si el carácter es un espacio | (EN) If the character is a space
12            e += 1
13        if ord(i) >= 97 and ord(i) <= 122: # (ES) Si está entre 'a' (97) y 'z' (122) es minúscula | (EN) If between 'a' (97) and 'z' (122), it's lowercase
14            min += 1
15        if ord(i) >= 65 and ord(i) <= 90: # (ES) Si está entre 'A' (65) y 'Z' (90) es mayúscula | (EN) If between 'A' (65) and 'Z' (90), it's uppercase
16            m += 1
17
18    print(f'Los números son: {n} \nLos espacios: {e} \nLas minúsculas: {min} \nLas mayúsculas: {m}')
19    # (ES) Muestra el total de números, espacios, minúsculas y mayúsculas | (EN) Displays totals of numbers, spaces, lowercase and uppercase
20
21 if __name__ == '__main__': # (ES) Punto de entrada del programa | (EN) Entry point of the program
22     inicio() # (ES) Ejecuta la función inicio | (EN) Runs the inicio function
23

```

Código programa 8

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python> & "C:/Us
cia/OneDrive/Escritorio/Practicas de Python/practicaspacial1/Programa8.py"
Escribe una cadena: Diego pc
Los numeros son: 0
Los espacios: 1
Las minusculas: 6
Las mayusculas: 1
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python>

```

Código ejecución 8





Programa 9: Hacer un programa que en una lista se introduzca cadenas de caracteres con las siguientes restricciones

1.- las cadenas no deben de tener espacio

2.- la cadena solo puede tener mayúsculas la primer letra

3.- obligatoriamente debe de tener todas las vocales.

El programa no termina hasta que la lista tenga 5 elementos

```

● ● ●
1 def vocales(cad):
2     a = False
3     e = False
4     i = False
5     o = False
6     u = False
7     #for i in cad:
8     if 'a' in cad or 'A' in cad:
9         a = True
10    if 'e' in cad or 'E' in cad:
11        e = True
12    if 'i' in cad or 'I' in cad:
13        i = True
14    if 'o' in cad or 'O' in cad:
15        o = True
16    if 'u' in cad or 'U' in cad:
17        u = True
18    if a == True and e == True and i == True and o == True and u == True:
19        lista.append(cad)
20        print(lista)
21    else:
22        print('Error')

```

Código programa 9

```

● ● ●
1 def minusculas(m):
2     cm = 0
3     pass
4     print(m)
5     for i in m[1: ]:
6         if ord(i) >= 97 and ord(i) <= 122:
7             cm +=1
8     if cm == len(m)-1:
9         print(f'la cadena son minusculas escepto la primera{cm}')
10        vocales(m)
11    else:
12        print('Error la cadena no cumple')

```

Código programa 9





```

● ● ●

1 def leer():
2     ce = 0
3     nc = ""
4     c = input('Ingresa una cadena \n')
5     for i in c:
6         if ord(i) != 32:
7             ce += 1
8         if ce == len(c):
9             if c.isalpha():
10                 #revisamos que sea mayusculas
11                 minusculas(c)
12             else:
13                 for i in c:
14                     if ord(i) >= 48 and ord(i) <= 57:
15                         pass
16                     else:
17                         nc += i
18                 print(nc)
19                 minusculas(nc)
20         else:
21             print("Error la cadena no completa")

```

Código programa 9

```

● ● ●

1 lista = []
2 if __name__ == '__main__':
3     while(True):
4         leer()
5         if len(lista) >= 5:
6             break

```

Código programa 9

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\OneDrive\Escritorio\Programas sin comentar\F
Ingresa una cadena
Diegoaeiou22
Diegoaeiou
Diegoaeiou
Diegoaeiou
la cadena son minusculas esepeto la primera9
['Diegoaeiou']
Ingresa una cadena

```

Código ejecución 9





Tare1: Hacer un programa que lea 10 datos, si el dato es un número se almacenara en un arreglo si es un carácter o caracteres se meterá a una lista, cuando finalice el programa nos mostrará cuantos elementos números y cuantos caracteres hay en cada estructura.

```

1 # (ES) Explicación del programa | (EN) Program description
2
3 d = 0 # (ES) Contador de datos ingresados | (EN) Counter of entered data
4 a = [0,0,0,0,0,0,0,0,0] # (ES) Arreglo inicializado con 10 ceros | (EN) Array initialized with 10 zeros
5 b = [] # (ES) Lista vacía para guardar letras | (EN) Empty list to store letters
6
7 while(d <= 9): # (ES) Bucle hasta que se ingresen 10 datos válidos | (EN) Loop until 10 valid inputs are entered
8     dato = input('Ingresa algun numero o letras: ') # (ES) Solicitud al usuario un dato | (EN) Asks the user for a value
9     if dato.isdigit(): # (ES) Si el dato son solo digitos | (EN) If the input is only digits
10         a.append(dato) # (ES) Agrega el número a la lista 'a' | (EN) Appends the number to 'a'
11         d += 1 # (ES) Incrementa el contador de datos válidos | (EN) Increments valid data counter
12     elif dato.isalpha(): # (ES) Si el dato son solo letras | (EN) If the input is only letters
13         b.append(dato) # (ES) Agrega la letra a la lista 'b' | (EN) Appends the letter to 'b'
14         d += 1 # (ES) Incrementa el contador de datos válidos | (EN) Increments valid data counter
15     else:
16         print('No es valido') # (ES) Mensaje si el dato no es número ni letra | (EN) Message if input is not number or letter
17
18 print(f'Estos son tus numeros ingresados {a}') # (ES) Muestra los números | (EN) Shows the numbers
19 print(f'Estas son las letras ingresadas{b}')

```

Código programa 10

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python\practicaspacial1> & "C:/Users/Diego Plascencia/OneDrive/Escritorio/Practicas de Python/practicaspacial1/Tarea1.py"
Ingresa algun numero o letras: diego
Ingresa algun numero o letras: pc
Ingresa algun numero o letras: 78
Ingresa algun numero o letras: pefo
Ingresa algun numero o letras: 778as
No es valido
Ingresa algun numero o letras: ad
Ingresa algun numero o letras:asd
Ingresa algun numero o letras:asd
Ingresa algun numero o letras: 754
Ingresa algun numero o letras: 57
Ingresa algun numero o letras:asd
Estos son tus numeros ingresados [0, 0, 0, 0, 0, 0, 0, 0, 0, '78', '754', '57']
Estas son las letras ingresadas['diego', 'pc', 'pefo', 'ad', 'asd', 'asd', 'asd']
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python\practicaspacial1>

```

Código ejecución 10





Tarea2: Siguiendo los mismos planteamientos del programa9

```

1 def vocales(v): # (ES) Verifica que la cadena tenga todas las vocales | (EN) Checks if string contains all vowels
2     va = False
3     ve = False
4     vi = False
5     vo = False
6     vu = False
7     if 'a' in v or 'A' in v:
8         va = True
9     if 'e' in v or 'E' in v:
10        ve = True
11    if 'i' in v or 'I' in v:
12        vi = True
13    if 'o' in v or 'O' in v:
14        vo = True
15    if 'u' in v or 'U' in v:
16        vu = True
17    if va == True and ve == True and vi == True and vo == True and vu == True: # (ES) Si todas las vocales están presentes | (EN) If all vowels are present
18        lista.append(v) # (ES) Agrega la cadena a la lista | (EN) Appends string to the list
19        print(lista) # (ES) Muestra la lista actual | (EN) Prints current list
20    else:
21        print('No contiene las vocales') # (ES) Mensaje si no contiene todas las vocales | (EN) Message if string doesn't have all vowels
22
23
24 def minusculas(m): # (ES) Verifica que todo, excepto la primera letra, sea minúscula | (EN) Checks that all except first letter are lowercase
25     cm = 0 # (ES) Contador de minusculas | (EN) lowercase counter
26     for i in m[1:]:
27         if ord(i) >= 97 and ord(i) <= 122: # (ES) Es minúscula | (EN) Is lowercase
28             cm += 1
29     if cm == len(m)-1: # (ES) Si todas son minúsculas menos la primera | (EN) If all lowercase except first
30         print('Son minusculas, excepto la primera: ',m)
31         vocales(m) # (ES) Llama a la función de vocales | (EN) Calls vowel check function
32     else:
33         print('Se encontro más de una mayúscula') # (ES) Si hay más de una mayúscula | (EN) More than one uppercase found
34
35
36 def pedir(): # (ES) Función principal para pedir cadenas | (EN) Main function to request strings
37     while(len(lista) <= 4): # (ES) Mientras la lista tenga meno de 5 elementos | (EN) While list has less than 5 elements
38         ca = "" # (ES) Variable para filtrar caracteres no deseados | (EN) Variable to filter unwanted characters
39         cad = input('Ingresa una cadena \n') # (ES) Solicita cadena al usuario | (EN) Ask user for string
40         for i in cad:
41             if i == " ": # (ES) Detecta espacios | (EN) Detect spaces
42                 print('Error, espacio') # (ES) Mensaje de error | (EN) Error message
43             if cad.isalpha(): # (ES) Si solo hay letras | (EN) If only letters
44                 minusculas(cad) # (ES) Revisa mayúscula/minúscula | (EN) Check uppercase/lowercase
45             else: # (ES) Si hay números u otros caracteres | (EN) If numbers or other characters
46                 for i in cad:
47                     if ord(i) >= 48 and ord(i) <= 57: # (ES) Ignora números | (EN) Ignore numbers
48                         pass
49                     else:
50                         ca += i # (ES) Agrega caracteres válidos | (EN) Append valid characters
51             minusculas(ca) # (ES) Revisa mayúscula/minúscula | (EN) Check uppercase/lowercase
52
53
54 lista = [] # (ES) Lista para almacenar cadenas válidas | (EN) List to store valid strings
55 if __name__ == '__main__':
56     pedir() # (ES) Llama a la función principal | (EN) Calls main function

```

Código programa 11

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial1> & "C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial1\Tarea2.py"
Ingresa una cadena
diegopc22aeiou
Son minusculas, excepto la primera: diegopcaeiu
['diegopcaeiu']
Ingresa una cadena
diego22
Son minusculas, excepto la primera: diego
No contiene las vocales
Ingresa una cadena

```

Código ejecución 11





Repaso1: Hacer un programa que lea nombre y precio de un producto el programa calculará el costo y precio de venta.

```

1 def costo(): # (ES) Función principal que calcula los precios | (EN) Main function that calculates prices
2     while(True): # (ES) Bucle infinito hasta que el usuario decida salir | (EN) Infinite loop until user decides to exit
3         cos = 0 # (ES) Variable para el costo | (EN) Variable for cost
4         iva = 0 # (ES) Variable para el precio final con IVA | (EN) Variable for final price with VAT
5         nom = input('¿Cuál es el nombre de tu producto? \n') # (ES) Pide el nombre del producto | (EN) Asks for product name
6         pre = float(input('¿Cuál es el precio de tu producto? \n')) # (ES) Pide el precio y convierte a float | (EN) Asks for price and converts to float
7         cos = float(pre * 0.12 + pre) # (ES) Calcula el costo sumando 12% | (EN) Calculates cost adding 12%
8         iva = float(pre * 0.16) # (ES) Calcula el IVA 16% sobre el precio original | (EN) Calculates 16% VAT on original price
9         iva = iva + cos # (ES) Precio final sumando IVA al costo | (EN) Final price adding VAT to cost
10        print(f'El precio final de {nom} es ${cos:.2f}') # (ES) Muestra el costo | (EN) Displays the cost
11        print(f'El precio final de {nom} es ${iva:.2f}') # (ES) Muestra el precio final con IVA | (EN) Displays final price with VAT
12        res = input('Deseas terminar este programa (s/n) \n') # (ES) Pregunta si desea terminar | (EN) Asks if user wants to exit
13        if res == 's' or res == 'S': # (ES) Si responde sí, rompe el bucle | (EN) If yes, breaks loop
14            break
15
16 if __name__ == "__main__": # (ES) Punto de entrada del programa | (EN) Program entry point
17     costo() # (ES) Llama a la función costo | (EN) Calls the cost function

```

Código programa 12

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python\practicaspacial1> & "C:/Users/Diego Plascencia/OneDrive/Escritorio/Practicas de Python/practicaspacial1/Repaso1.py"
¿Cuál es el nombre de tu producto?
coa
¿Cuál es el precio de tu producto?
78
El costo final de coa es $87.36
El precio final de coa es $99.84
Deseas terminar este programa (s/n)
n
¿Cuál es el nombre de tu producto?
pepino
¿Cuál es el precio de tu producto?
10
El costo final de pepino es $11.20
El precio final de pepino es $12.80
Deseas terminar este programa (s/n)
s
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python\practicaspacial1>

```

Código ejecución 12





Repaso2: Fórmula general para ecuaciones cuadráticas

```

● ● ●
1 a = 1 # (ES) Coeficiente de x^2 | (EN) Coefficient of x^2
2 b = 2 # (ES) Coeficiente de x | (EN) Coefficient of x
3 c = -15 # (ES) Término independiente | (EN) Constant term
4
5 p = 0 # (ES) Variable temporal para b^2 | (EN) Temporary variable for b^2
6 m = 0 # (ES) Variable temporal para 4*a*c | (EN) Temporary variable for 4*a*c
7 r = 0.0 # (ES) Discriminante | (EN) Discriminant
8 d = 0.0 # (ES) Denominador 2*a | (EN) Denominator 2*a
9 x1 = 0.0 # (ES) Raíz 1 | (EN) Root 1
10 x2 = 0.0 # (ES) Raíz 2 | (EN) Root 2
11
12 p = b ** 2 # (ES) b^2 | (EN) b squared
13 m = 4 * a * c # (ES) 4*a*c | (EN) 4*a*c
14 r = p - m # (ES) Discriminante: b^2 - 4*a*c | (EN) Discriminant: b^2 - 4*a*c
15
16 if r > 0: # (ES) Si el discriminante es positivo, hay dos soluciones reales | (EN) If discriminant is positive, there are two real solutions
17     print('si se puede continuar') # (ES) Indica que se puede calcular | (EN) Indicates calculation can proceed
18     ra = r**(1/2) # (ES) Raíz cuadrada del discriminante | (EN) Square root of discriminant
19     d = 2*a # (ES) Denominador de la fórmula general | (EN) Denominator of the quadratic formula
20     x1 = (-b + ra)/d # (ES) Primera raíz | (EN) First root
21     x2 = (-b - ra)/d # (ES) Segunda raíz (X error: debe ser -ra) | (EN) Second root (X mistake: should be -ra)
22     print(f'El valor de x1 es {x1:.2f} y de {x2:.2f}') # (ES) Muestra ambas raíces | (EN) Prints both roots
23 else:
24     print('no se puede continuar') # (ES) Si el discriminante es negativo, no hay raíces reales | (EN) If discriminant is negative, no real roots
25

```

Código programa 13

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial1> & 'sers/Diego Plascencia/OneDrive/Escritorio/Prácticas de Python/prácticasparcial1/Repaso2.py'
si se puede continuar
El valor de x1 es 3.00 y de 3.00
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial1>

```

Código ejecución 13





Repaso3: Hacer un programa que Lea un dato cual sea y que lo almacene en una lista, respetando su tipo de dato

```

1 def validar(dato):
2     f = 0.0
3     i = 0
4
5     try:
6         i = int(dato)
7         print('Es un entero')
8         return i
9     except ValueError:
10        print('No es un entero')
11
12     try:
13         f = float(dato)
14         print('Es un flotante')
15         return f
16     except ValueError:
17         print('No es un flotante')
18         print('Es un string')
19     return dato
20
21 def leer():
22     a = input('Ingrisa un dato \n')
23     dat = validar(a)
24     lista.append(dat)
25
26 lista = []
27 if __name__ == '__main__':
28     while(True):
29         leer()
30         r = input('deas añadir un nuevo dato S/N \n')
31         if r == 'N' or r =='\n':
32             print(lista)
33             break
34

```

Código programa 14

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python> python programa_14.py
Ingrisa un dato
28
Es un entero
deas añadir un nuevo dato S/N
S
Ingrisa un dato
74.5
No es un entero
Es un flotante
deas añadir un nuevo dato S/N
S
Ingrisa un dato
diego
No es un entero
No es un flotante
Es un string
deas añadir un nuevo dato S/N

```

Código ejecución 14





ExamenPasado: Programa interactivo que solicita datos al usuario, realiza comparaciones y muestra los resultados en pantalla. Probablemente reproduce ejercicios de un examen anterior.

```

● ● ●
1 def inicio():
2     lista1 = [] # Lista vacía para guardar los datos del usuario / Empty list to store user's data
3     nom = input('Cual es tu nombre') # Solicita el nombre del usuario / Asks for the user's name
4     c1 = int(input('Ingresa una calificación')) # Pide la primera calificación / Asks for the first grade
5     c2 = int(input('Ingresa una calificación')) # Pide la segunda calificación / Asks for the second grade
6     c3 = int(input('Ingresa una calificación')) # Pide la tercera calificación / Asks for the third grade
7
8     mayor = max(c1,c2,c3) # Encuentra la calificación más alta / Finds the highest grade
9     menor = min(c1,c2,c3) # Encuentra la calificación más baja / Finds the lowest grade
10
11    lista2 = [] # Lista vacía para guardar múltiples registros / Empty list to store multiple records
12    if __name__ == '__main__': # Comprueba si el programa se ejecuta directamente / Checks if the program runs directly
13        while(True): # Bucle infinito hasta que el usuario decida salir / Infinite loop until user decides to exit
14            inicio() # Llama a la función inicio() / Calls the inicio() function
15            a = input('Quieres añadir otro registro s/n') # Pregunta si desea añadir otro registro / Asks if the user wants to add another record
16            if a == 'N' or a == 'n': # Si el usuario responde 'n' o 'N', termina el ciclo / If user answers 'n' or 'N', ends the loop
17                print(lista2) # Muestra la lista de registros / Displays the list of records
18                break # Rompe el ciclo while / Breaks the while loop

```

Código programa 15

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial2> & "C:/Users/Diego
/Plascencia/OneDrive/Escritorio/Prácticas de Python/prácticasparcial2/Examenpasado.py"
Cual es tu nombreDiego
Ingresa una calificación100
Ingresa una calificación100
Ingresa una calificación95
Quieres añadir otro registro s/n
[]
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial2>

```

Código ejecución 15





Programa1: Similar al anterior: pide datos al usuario, ejecuta condiciones y muestra mensajes según los valores ingresados.

```

1 # Introduccion a la recursividad, como una funcion se llama asi misma dentro de la misma.
2 # Introduction to recursion, how a function calls itself within itself.
3
4 def inicio(num):
5     #global num # (Comentado) Se usaria si quisieramos modificar la variable global 'num' / Would be used if we wanted to modify the global variable 'num'
6     a = int(input('Escribe una calificacion \n')) # Pide al usuario ingresar una calificación / Asks the user to enter a grade
7     num += 1 # Incrementa el contador en 1 cada vez que se llama la función / Increases the counter by 1 each time the function is called
8     lista.append(a) # Agrega la calificación ingresada a la lista / Adds the entered grade to the list
9     if num >= 5: # Condición base: cuando se han ingresado 5 calificaciones / Base condition: when 5 grades have been entered
10        print(lista) # Muestra todas las calificaciones guardadas / Displays all stored grades
11    else:
12        #return inicio(num) # (Comentado) Podria usarse para devolver el resultado de la llamada recursiva / Could be used to return the result of the recursive call
13        inicio(num) # Llamada recursiva: la función se llama a si misma / Recursive call: the function calls itself
14
15
16 lista = [] # Lista vacía donde se almacenarán las calificaciones / Empty list where grades will be stored
17 global num # Declaración de variable global 'num' / Declaration of global variable 'num'
18 num = 0 # Inicializa el contador en 0 / Initializes the counter at 0
19 if __name__ == '__main__': # Verifica si el archivo se ejecuta directamente / Checks if the file is being run directly
20     inicio(num) # Llama a la función 'inicio' para comenzar el proceso / Calls the 'inicio' function to start the process
21

```

Código programa 16

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial2> & "C:/Users/Diego Plascencia/AppData/Roaming/Microsoft/Windows/Start Menu/Programs/Python 3.8/Python.exe" "C:/Users/Diego Plascencia/OneDrive/Escritorio/Prácticas de Python/prácticasparcial2/Programa1.py"
Escribe una calificación
100
Escribe una calificación
78
Escribe una calificación
45
Escribe una calificación
78
Escribe una calificación
10
[100, 78, 45, 78, 10]
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Prácticas de Python\prácticasparcial2>

```

Código ejecución 16





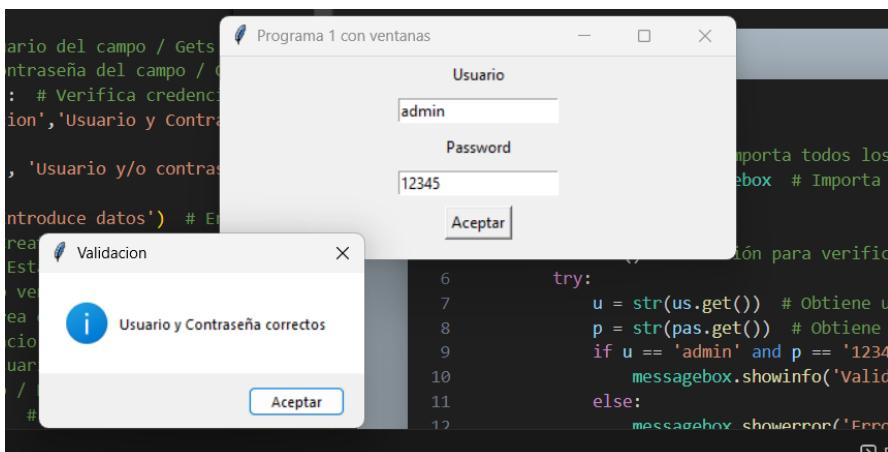
Practica2: Usa una interfaz gráfica con Tkinter, permitiendo interactuar mediante ventanas y botones en lugar de la consola.

```

1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
3
4  def Ventana():
5      def revisar(): # Función para verificar credenciales / Function to check credentials
6          try:
7              u = str(us.get()) # Obtiene usuario del campo / Gets username from field
8              p = str(pas.get()) # Obtiene contraseña del campo / Gets password from field
9              if u == 'admin' and p == '12345': # Verifica credenciales / Checks credentials
10                  messagebox.showinfo('Validacion','Usuario y Contraseña correctos') # Mensaje de éxito / Success message
11              else:
12                  messagebox.showerror('Error', 'Usuario y/o contraseña incorrectos') # Mensaje de error / Error message
13          except ValueError:
14              messagebox.showerror('Error', 'Introduce datos') # Error si no hay datos / Error if no data
15  ven = Tk() # Crea ventana principal / Creates main window
16  ven.title('Programa 1 con ventanas') # Establece título / Sets title
17  ven.geometry('400x200') # Define tamaño ventana / Defines window size
18  label = Label(ven, text='Usuario') # Crea etiqueta usuario / Creates username label
19  label.pack(pady=4) # Empaquetá con espacio / Packs with spacing
20  us = Entry(ven) # Crea campo entrada usuario / Creates username entry field
21  us.pack(pady=4) # Empaquetá con espacio / Packs with spacing
22  Label(ven, text='Password').pack(pady=4) # Etiqueta contraseña y empaquetá / Password label and packs
23  pas = Entry(ven) # Crea campo entrada contraseña / Creates password entry field
24  pas.pack(pady=4) # Empaquetá con espacio / Packs with spacing
25  boton = Button(ven, text='Aceptar',command=revisar).pack(pady=4) # Botón que ejecuta revisar / Button that runs check
26  ven.mainloop() # Inicia bucle principal / Starts main loop
27
28 if __name__ == '__main__':
29     Ventana() # Ejecuta la función Ventana / Runs Ventana function

```

Código programa 17



Código ejecución 17





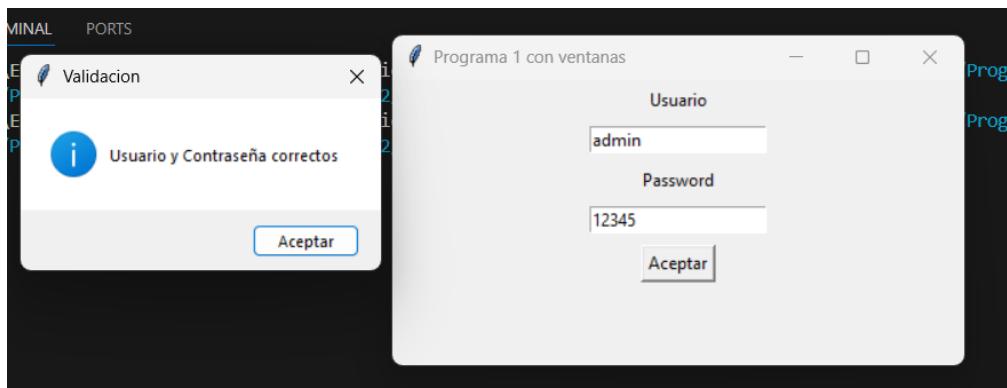
Programa3: También emplea **Tkinter**, posiblemente como segunda práctica con GUI, quizá mejorando el diseño o la funcionalidad del Programa2.

```

1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
3
4  def Ventana():
5      def revisar(): # Función para verificar credenciales / Function to check credentials
6          try:
7              u = str(us.get()) # Obtiene usuario del campo / Gets username from field
8              p = str(pas.get()) # Obtiene contraseña del campo / Gets password from field
9              if u == 'admin' and p == '12345': # Verifica credenciales / Checks credentials
10                  messagebox.showinfo('Validacion','Usuario y Contraseña correctos') # Mensaje de éxito / Success message
11              else:
12                  messagebox.showerror('Error', 'Usuario y/o contraseña incorrectos') # Mensaje de error / Error message
13          except ValueError:
14              messagebox.showerror('Error', 'Introduce datos') # Error si no hay datos / Error if no data
15  ven = Tk() # Crea ventana principal / Creates main window
16  ven.title('Programa 1 con ventanas') # Establece título / Sets title
17  ven.geometry('400x200') # Define tamaño ventana / Defines window size
18  label = Label(ven, text='Usuario') # Crea etiqueta usuario / Creates username label
19  label.pack(pady=4) # Empaquetá con espacio / Packs with spacing
20  us = Entry(ven) # Crea campo entrada usuario / Creates username entry field
21  us.pack(pady=4) # Empaquetá con espacio / Packs with spacing
22  Label(ven, text='Password').pack(pady=4) # Etiqueta contraseña y empaquetá / Password label and packs
23  pas = Entry(ven) # Crea campo entrada contraseña / Creates password entry field
24  pas.pack(pady=4) # Empaquetá con espacio / Packs with spacing
25  boton = Button(ven, text='Aceptar',command=revisar).pack(pady=4) # Botón que ejecuta revisar / Button that runs check
26  ven.mainloop() # Inicia bucle principal / Starts main loop
27
28 if __name__ == '__main__':
29     Ventana() # Ejecuta la función Ventana / Runs Ventana function

```

Código programa 18



Código ejecución 18





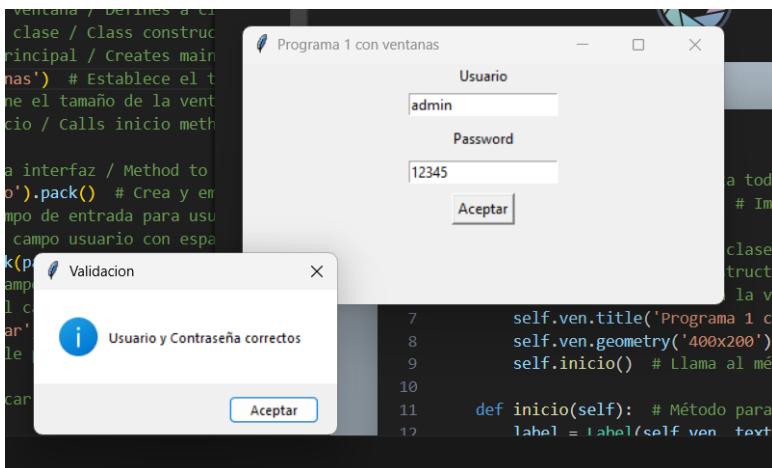
Programa4: Utiliza **ciclos (for o while)** para repetir acciones; puede manejar listas o crear menús con repeticiones controladas.

```

1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
3
4  class Ventana(): # Define una clase para la ventana / Defines a class for the window
5      def __init__(self): # Constructor de la clase / Class constructor
6          self.ven = TK() # Crea la ventana principal / Creates main window
7          self.ven.title('Programa 1 con ventanas') # Establece el título / Sets the title
8          self.ven.geometry('400x200') # Define el tamaño de la ventana / Defines window size
9          self.inicio() # Llama al método inicio / Calls inicio method
10
11     def inicio(self): # Método para crear la interfaz / Method to create interface
12         label = Label(self.ven, text='Usuario').pack() # Crea y empaqueta etiqueta usuario / Creates and packs username label
13         self.us = Entry(self.ven) # Crea campo de entrada para usuario / Creates username entry field
14         self.us.pack(pady=4) # Empaqueña el campo usuario con espacio / Packs username field with spacing
15         Label(self.ven, text='Password').pack(pady=4) # Crea y empaqueta etiqueta contraseña / Creates and packs password label
16         self.pas = Entry(self.ven) # Crea campo de entrada para contraseña / Creates password entry field
17         self.pas.pack(pady=4) # Empaqueña el campo contraseña con espacio / Packs password field with spacing
18         boton = Button(self.ven, text='Aceptar', command=self.revisar).pack(pady=4) # Crea y empaqueta botón / Creates and packs button
19         self.ven.mainloop() # Inicia el bucle principal / Starts main loop
20
21     def revisar(self): # Método para verificar credenciales / Method to check credentials
22         try:
23             u = str(self.us.get()) # Obtiene el texto del campo usuario / Gets text from username field
24             p = str(self.pas.get()) # Obtiene el texto del campo contraseña / Gets text from password field
25             if u == 'admin' and p == '12345': # Verifica las credenciales / Checks credentials
26                 messagebox.showinfo('Validacion', 'Usuario y Contraseña correctos') # Mensaje de éxito / Success message
27             else:
28                 messagebox.showerror('Error', 'Usuario y/o contraseña incorrectos') # Mensaje de error / Error message
29         except ValueError:
30             messagebox.showerror('Error', 'Introduce datos') # Error si no hay datos válidos / Error if no valid data
31
32
33 if __name__ == '__main__':
34     app = Ventana() # Crea una instancia de la clase Ventana / Creates an instance of Ventana class

```

Código programa 19



Código ejecución 19





Programa5: Define funciones propias que devuelven resultados. Parece centrado en cálculos o validaciones específicas.

```

1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
3
4  class Principal(): # Define la clase principal / Defines main class
5      def __init__(self): # Constructor de la clase / Class constructor
6          self.ven = Tk() # Crea la ventana principal / Creates main window
7          self.ven.title('Programa 5 con ventanas') # Establece el título / Sets the title
8          self.ven.geometry('450x250') # Define el tamaño de la ventana / Defines window size
9          self.lista = [] # Inicializa lista vacía para almacenar promedios / Initializes empty list to store averages
10         self.inicio() # Llama al método inicio / Calls inicio method
11
12     def sumar(self): # Método para sumar elementos de la lista / Method to sum list elements
13         s = 0 # Inicializa contador de suma / Initializes sum counter
14         for i in self.lista: # Itera sobre cada elemento de la lista / Iterates over each list element
15             s += i # Suma el elemento actual al total / Adds current element to total
16         return s # Retorna la suma total / Returns total sum
17
18     def promediar(self): # Método para calcular promedio / Method to calculate average
19         try:
20             a = float(self.n1.get()) # Obtiene y convierte primer número / Gets and converts first number
21             b = float(self.n2.get()) # Obtiene y convierte segundo número / Gets and converts second number
22             c = float(self.n3.get()) # Obtiene y convierte tercer número / Gets and converts third number
23             d = float(self.n4.get()) # Obtiene y convierte cuarto número / Gets and converts fourth number
24             promedio = (a+b+c+d) / 4 # Calcula el promedio / Calculates average
25             self.16.config(text = str(promedio)) # Actualiza etiqueta con promedio / Updates label with average
26             self.lista.append(promedio) # Agrega promedio a la lista / Adds average to list
27             self.17.config(text = str(self.lista)) # Actualiza etiqueta con lista / Updates label with list
28             self.n1.delete(0,END) # Limpia campo de entrada 1 / Clears entry field 1
29             self.n2.delete(0,END) # Limpia campo de entrada 2 / Clears entry field 2
30             self.n3.delete(0,END) # Limpia campo de entrada 3 / Clears entry field 3
31             self.n4.delete(0,END) # Limpia campo de entrada 4 / Clears entry field 4
32             suma = self.sumar() # Calcula suma total de promedios / Calculates total sum of averages
33             p = suma / len(self.lista) # Calcula promedio general / Calculates general average
34             self.18.config(text = f'Promedio general: {str(p)}') # Actualiza promedio general / Updates general average
35         except ValueError: # Captura error de conversión / Catches conversion error
36             messagebox.showerror('Error', 'Algun dato no es un numero') # Muestra mensaje de error / Shows error message
37             self.n1.delete(0,END) # Limpia campo de entrada 1 / Clears entry field 1
38             self.n2.delete(0,END) # Limpia campo de entrada 2 / Clears entry field 2
39             self.n3.delete(0,END) # Limpia campo de entrada 3 / Clears entry field 3
40             self.n4.delete(0,END) # Limpia campo de entrada 4 / Clears entry field 4
41
42     def salir(self): # Método para cerrar la aplicación / Method to close application
43         self.ven.destroy() # Cierra la ventana / Closes window
44
45     def inicio(self): # Método para crear la interfaz / Method to create interface
46         l1 = Label(self.ven,text='Escribe un numero').place(y=10,x=20) # Etiqueta y posición / Label and position
47         l2 = Label(self.ven,text='Escribe un numero').place(y=50,x=20) # Etiqueta y posición / Label and position
48         self.n1 = Entry(self.ven) # Campo de entrada para número 1 / Entry field for number 1
49         self.n1.place(y=10,x=130) # Posición del campo 1 / Position of field 1
50         self.n2 = Entry(self.ven) # Campo de entrada para número 2 / Entry field for number 2
51         self.n2.place(y=50,x=130) # Posición del campo 2 / Position of field 2
52         l3 = Label(self.ven,text='Escribe un numero').place(y=90,x=20) # Etiqueta y posición / Label and position
53         l4 = Label(self.ven,text='Escribe un numero').place(y=130,x=20) # Etiqueta y posición / Label and position
54         self.n3 = Entry(self.ven) # Campo de entrada para número 3 / Entry field for number 3
55         self.n3.place(y=90,x=130) # Posición del campo 3 / Position of field 3
56         self.n4 = Entry(self.ven) # Campo de entrada para número 4 / Entry field for number 4
57         self.n4.place(y=130,x=130) # Posición del campo 4 / Position of field 4
58         l5 = Label(self.ven,text='Promedio').place(y=150,x=130) # Etiqueta promedio / Average label
59         self.16 = Label(self.ven,text='0.0') # Etiqueta para mostrar promedio actual / Label to show current average
60         self.16.place(y=150,x=200) # Posición etiqueta promedio / Average label position
61         b1 = Button(self.ven,text='Promedio',command=self.promediar).place(y=50,x=300) # Botón calcular promedio / Calculate average button
62         b2 = Button(self.ven,text='Salir',command=self.salir).place(y=90,x=300) # Botón salir / Exit button
63         self.17 = Label(self.ven,text='[]') # Etiqueta para mostrar lista de promedios / Label to show averages list
64         self.17.place(y=170,x=200) # Posición etiqueta lista / List label position
65         self.18 = Label(self.ven,text='Promedio general: 0.0') # Etiqueta promedio general / General average label
66         self.18.place(y=190,x=130) # Posición promedio general / General average position
67         self.ven.mainloop() # Inicia el bucle principal / Starts main loop
68
69     if __name__ == '__main__':
70         app = Principal() # Crea instancia de la clase Principal / Creates instance of Principal class

```

Código programa 20





Programa 5 con ventanas

Escribe un numero

Escribe un numero

Escribe un numero

Escribe un numero

Promedio

Salir

Promedio 5.5
[5.5]
Promedio general: 5.5

Código ejecución 20





Programa6: Estructura modular con funciones reutilizables, posiblemente una continuación o mejora del Programa5.

```

1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
3
4  class Principal(): # Define la clase principal / Defines main class
5      def __init__(self): # Constructor de la clase / Class constructor
6          self.ven = Tk() # Crea la ventana principal / Creates main window
7          self.ven.title('Programa 6 con ventanas GRID') # Establece el título / Sets the title
8          self.ven.geometry('450x350') # Define el tamaño de la ventana / Defines window size
9          self.a = 0 # Variable para primer número / Variable for first number
10         self.b = 0 # Variable para segundo número / Variable for second number
11         self.lista = [] # Lista para almacenar números / List to store numbers
12         self.aux1 = 0 # Variable auxiliar para mayor / Auxiliary variable for maximum
13         self.aux2 = 0 # Variable auxiliar para menor / Auxiliary variable for minimum
14         self.cont = 0 # Contador para recorrer lista / Counter to traverse list
15
16     def Inicio(self): # Método para crear la interfaz / Method to create interface
17         l1 = Label(self.ven,text="Programa 6") # Etiqueta título / Title label
18         l1.grid(row=1,column=2) # Posición en grid / Grid position
19         l2 = Label(self.ven,text="Escribe un numero") # Etiqueta para primer número / Label for first number
20         l2.grid(row=3,column=1,padx=5,pady=5) # Posición con padding / Position with padding
21         Label(self.ven,text="").grid(row=2,column=2) # Espacio vacío / Empty space
22         self.n1 = Entry(self.ven) # Campo entrada primer número / Entry field first number
23         self.n1.grid(row=3,column=2) # Posición en grid / Grid position
24         l3 = Label(self.ven,text="Escribe otro numero") # Etiqueta para segundo número / Label for second number
25         l3.grid(row=5,column=1,padx=5,pady=5) # Posición con padding / Position with padding
26         Label(self.ven,text="").grid(row=4,column=2) # Espacio vacío / Empty space
27         self.n2 = Entry(self.ven) # Campo entrada segundo número / Entry field second number
28         self.n2.grid(row=5,column=2) # Posición en grid / Grid position
29         b1 = Button(self.ven,text="Agregar",command=self.agregar) # Botón agregar números / Add numbers button
30         b1.grid(row=6,column=1,padx=10) # Posición con padding / Position with padding
31         b2 = Button(self.ven,text="Mayor",command=self.mayor) # Botón encontrar mayor / Find maximum button
32         b2.grid(row=6,column=2) # Posición en grid / Grid position
33         b3 = Button(self.ven,text="Menor",command=self.menor) # Botón encontrar menor / Find minimum button
34         b3.grid(row=6,column=3,padx=10) # Posición con padding / Position with padding
35         b4 = Button(self.ven,text="Salir",command=self.salir) # Botón salir / Exit button
36         b4.grid(row=6,column=4,padx=25) # Posición con padding / Position with padding
37         self.listaview = Label(self.ven,text="[]") # Etiqueta para mostrar lista / Label to show list
38         self.listaview.grid(row=8,column=2,padx=25) # Posición con padding / Position with padding
39         self.listView = Listbox(self.ven, height=10, width=15, bg="grey", activestyle='dotbox', fg='black') # Listbox para mostrar elementos / Listbox to show elements
40         self.listView.grid(row=2,column=4) # Posición en grid / Grid position
41         self.ven.mainloop() # Inicia el bucle principal / Starts main loop
42
43     def salir(self): # Método para cerrar la aplicación / Method to close application
44         self.ven.destroy() # Cierra la ventana / Closes window
45
46     def agregar(self): # Método para agregar números a la lista / Method to add numbers to list
47         try:
48             self.a = int(self.n1.get()) # Obtiene y convierte primer número / Gets and converts first number
49             self.lista.append(self.a) # Agrega a la lista / Adds to list
50             self.listView.insert(self.listView.size()+1,self.a) # Inserta en Listbox / Inserts in Listbox
51             self.n1.delete(0,END) # Limpia campo de entrada / Clears entry field
52             self.b = int(self.n2.get()) # Obtiene y convierte segundo número / Gets and converts second number
53             self.listView.insert(self.listView.size(),self.b) # Inserta en Listbox / Inserts in Listbox
54             self.lista.append(self.b) # Agrega a la lista / Adds to list
55             self.n2.delete(0,END) # Limpia campo de entrada / Clears entry field
56             print(self.lista) # Imprime lista en consola / Prints list to console
57             self.aux2 = self.lista[0] # Inicializa variable para menor / Initializes variable for minimum
58             self.listaElementos.config(text = f'{self.lista}') # Actualiza etiqueta lista / Updates list label
59
60         except ValueError: # Captura error de conversión / Catches conversion error
61             messagebox.showerror('Error','Agrega algún numero') # Muestra mensaje de error / Shows error message
62
63     def mayor(self): # Método para encontrar el número mayor / Method to find maximum number
64         if len(self.lista) > 0: # Verifica si la lista no está vacía / Checks if list is not empty
65             if self.aux1 < self.lista[self.cont]: # Compara con elemento actual / Compares with current element
66                 self.aux1 = self.lista[self.cont] # Actualiza el mayor / Updates maximum
67             self.cont += 1 # Incrementa contador / Increments counter
68             if len(self.lista)-1 < self.cont: # Verifica si llegó al final / Checks if reached end
69                 print(f'El mayor es {self.aux1}') # Imprime resultado / Prints result
70                 messagebox.showinfo('El mayor',self.aux1) # Muestra mensaje informativo / Shows info message
71                 self.cont = 0 # Reinicia contador / Resets counter
72             else:
73                 return self.mayor() # Llama recursivamente / Calls recursively
74         else:
75             messagebox.showerror('Error','La lista esta vacia') # Muestra error si lista vacía / Shows error if empty list
76
77     def menor(self): # Método para encontrar el número menor / Method to find minimum number
78         if len(self.lista) > 0: # Verifica si la lista no está vacía / Checks if list is not empty
79             for i in self.lista: # Itera sobre cada elemento / Iterates over each element
80                 if self.aux2 > i: # Compara con elemento actual / Compares with current element
81                     self.aux2 = i # Actualiza el menor / Updates minimum
82             print(f'El menor es {self.aux2}') # Imprime resultado / Prints result
83             messagebox.showinfo('El menor',self.aux2) # Muestra mensaje informativo / Shows info message
84         else:
85             messagebox.showerror('Error','La lista esta vacia') # Muestra error si lista vacía / Shows error if empty list
86
87     if __name__ == '__main__':
88         app = Principal() # Crea instancia de la clase / Creates class instance
89         app.Inicio() # Llama al método Inicio / Calls Inicio method

```

Código programa 21





Programa 6 con ventanas GRID

Programa 6

78
45

Escribe un numero

Escribe otro numero

[78, 45]

Código ejecución 21

Programa 6 con ventanas GRID

Programa 6

El mayor X

i 78

Aceptar

78
45

Escribe un numero

Escribe otro numero

[78, 45]

Código ejecución 21

Programa 6 con ventanas GRID

Programa 6

El menor X

i 45

Aceptar

78
45

Escribe un numero

Escribe otro numero

[78, 45]

Código ejecución 21





Programa7: Similar a los dos anteriores: implementa funciones que organizan la lógica y los cálculos de manera estructurada.

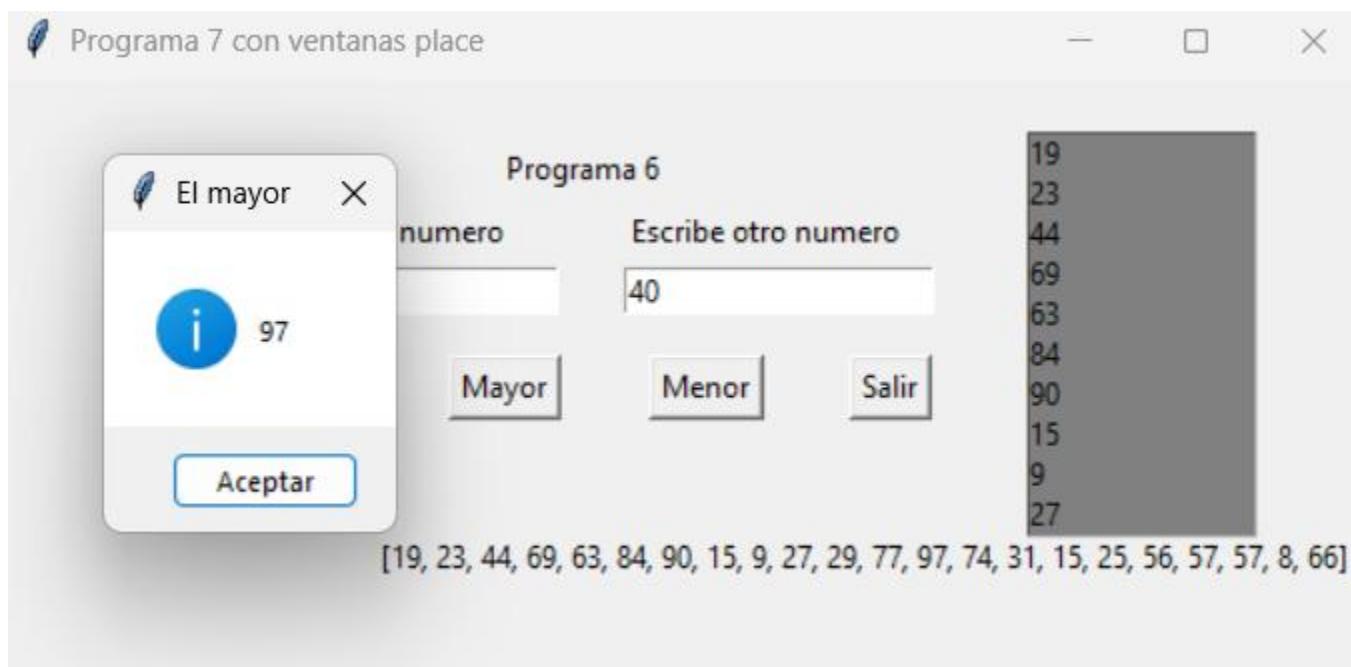
```

 1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
 2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
 3  import random # Importa módulo para números aleatorios / Imports module for random numbers
 4
 5  class Principal(): # Define la clase principal / Defines main class
 6      def __init__(self): # Constructor de la clase / Class constructor
 7          self.ven = Tk() # Crea la ventana principal / Creates main window
 8          self.ven.title('Programa 7 con ventanas place') # Establece el título / Sets the title
 9          self.ven.geometry('256x256') # Define el tamaño de la ventana / Defines window size
10          self.a = 0 # Variable para primer número / Variable for first number
11          self.b = 0 # Variable para segundo número / Variable for second number
12          self.lista = [] # Lista para almacenar números / List to store numbers
13          self.aux1 = 0 # Variable auxiliar para mayor / Auxiliary variable for maximum
14          self.aux2 = 0 # Variable auxiliar para menor / Auxiliary variable for minimum
15          self.cont = 0 # Contador para recorrer lista / Counter to traverse list
16
17      def Inicio(self): # Método para crear la interfaz / Method to create interface
18          l1 = Label(self.ven,text="Programa 6") # Etiqueta título / Title label
19          l1.place(x=200,y=25) # Posición absoluta con place / Absolute position with place
20          l2 = Label(self.ven,text="Escribe un número") # Etiqueta para primer número / Label for first number
21          l2.place(x=100,y=50) # Posición absoluta con place / Absolute position with place
22          l3 = Label(self.ven,text=" ") # Espacio vacío / Empty space
23          l3.place(x=100,y=75) # Campo entrada primer número / Entry field first number
24          self.n1 = Entry(self.ven) # Campo entrada primer número / Entry field first number
25          self.n1.place(x=100,y=75) # Posición absoluta con place / Absolute position with place
26          l4 = Label(self.ven,text="Escribe otro número") # Etiqueta para segundo número / Label for second number
27          l4.place(x=250,y=50) # Posición absoluta con place / Absolute position with place
28          l5 = Label(self.ven,text=" ") # Espacio vacío / Empty space
29          l5.place(x=250,y=75) # Campo entrada segundo número / Entry field second number
30          self.n2 = Entry(self.ven) # Campo entrada segundo número / Entry field second number
31          self.n2.place(x=250,y=75) # Posición absoluta con place / Absolute position with place
32          b1 = Button(self.ven,text="Aregar",command=self.agregar) # Botón agregar números / Adds numbers button
33          b1.place(x=100,y=110) # Posición absoluta con place / Absolute position with place
34          b2 = Button(self.ven,text="Mayor",command=self.mayor) # Botón encontrar mayor / Find maximum button
35          b2.place(x=180,y=110) # Posición absoluta con place / Absolute position with place
36          b3 = Button(self.ven,text="Menor",command=self.menor) # Botón encontrar menor / Find minimum button
37          b3.place(x=260,y=110) # Posición absoluta con place / Absolute position with place
38          b4 = Button(self.ven,text="Salir",command=self.salir) # Botón salir / Exit button
39          b4.place(x=340,y=110) # Posición absoluta con place / Absolute position with place
40          self.listelementos = Label(self.ven,text="[]") # Etiqueta para mostrar lista / Label to show list
41          self.listelementos.place(x=150,y=180) # Posición absoluta con place / Absolute position with place
42          self.listView = Listbox(self.ven, height=10, width=15, bg='grey', activestyle='dotbox', fg='black') # Listbox para mostrar elementos / Listbox to show elements
43          self.listView.place(x=410,y=20) # Posición absoluta con place / Absolute position with place
44          self.ven.mainloop() # Inicia el bucle principal / Starts main loop
45
46      def salir(self): # Método para cerrar la aplicación / Method to close application
47          self.ven.destroy() # Cierra la ventana / Closes window
48
49      def agregar(self): # Método para agregar números aleatorios a la lista / Method to add random numbers to list
50          try:
51              valor = random.randint(1,100) # Genera número aleatorio entre 1-100 / Generates random number between 1-100
52              self.lista.append(valor) # Agrega a la lista / Adds to list
53              self.listView.insert(self.listView.size()+1,valor) # Inserta en Listbox / Inserts in Listbox
54              print(self.lista) # Imprime lista en consola / Prints list to console
55              self.aux = self.lista[0] # Inicializa variable para menor / Initializes variable for minimum
56              self.listaElementos.config(text = f'{self.lista}') # Actualiza etiqueta lista / Updates list label
57
58          except ValueError: # Captura error de conversión / Catches conversion error
59              messagebox.showerror("Error","Agrega algún número") # Muestra mensaje de error / Shows error message
60
61      def mayor(self): # Método para encontrar el número mayor / Method to find maximum number
62          if len(self.lista) > 0: # Verifica si la lista no está vacía / Checks if list is not empty
63              if self.aux1 < self.lista[self.cont]: # Compara con elemento actual / Compares with current element
64                  self.aux1 = self.lista[self.cont] # Actualiza el mayor / Updates maximum
65              self.cont += 1 # Incrementa contador / Increments counter
66              if len(self.lista)-1 < self.cont: # Verifica si llegó al final / Checks if reached end
67                  print(f'El mayor es {self.aux1}') # Imprime resultado / Prints result
68                  messagebox.showinfo('El mayor',self.aux1) # Muestra mensaje informativo / Shows info message
69                  self.cont = 0 # Reinicia contador / Resets counter
70              else:
71                  return self.mayor() # Llama recursivamente / Calls recursively
72          else:
73              messagebox.showerror('Error','La lista esta vacía') # Muestra error si lista vacía / Shows error if empty list
74
75      def menor(self): # Método para encontrar el número menor / Method to find minimum number
76          if len(self.lista) > 0: # Verifica si la lista no está vacía / Checks if list is not empty
77              for i in self.lista: # Itera sobre cada elemento / Iterates over each element
78                  if self.aux2 > i: # Compara con elemento actual / Compares with current element
79                      self.aux2 = i # Actualiza el menor / Updates minimum
80              print(f'El menor es {self.aux2}') # Imprime resultado / Prints result
81              messagebox.showinfo('El menor',self.aux2) # Muestra mensaje informativo / Shows info message
82          else:
83              messagebox.showerror('Error','La lista esta vacía') # Muestra error si lista vacía / Shows error if empty list
84
85      if __name__ == '__main__':
86          app = Principal() # Crea instancia de la clase / Creates class instance
87          app.Inicio() # Llama al método Inicio / Calls Inicio method

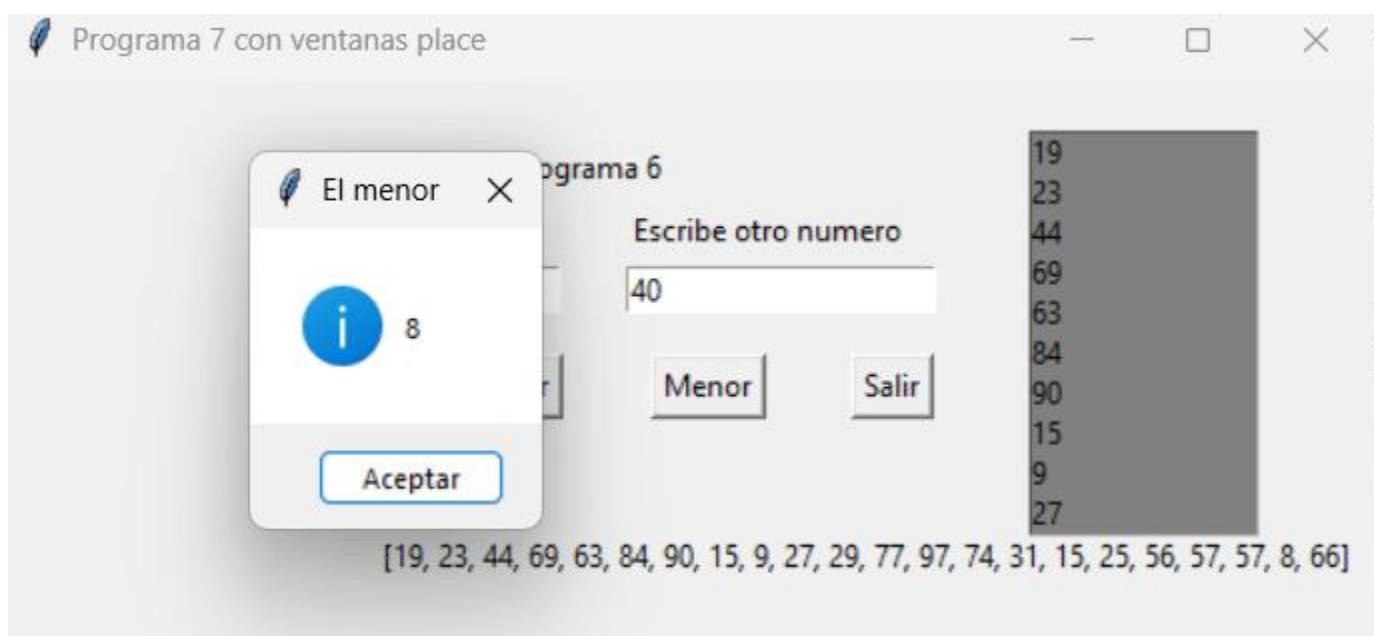
```

Código programa 22





Código ejecución 22



Código ejecución 22





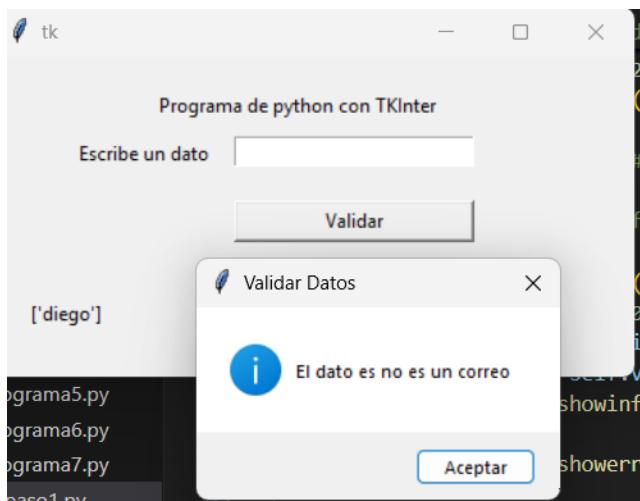
Repaso1: Incluye estructuras repetitivas para practicar bucles y condicionales; parece diseñado como ejercicio de repaso.

```

1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
3  from ValidarRepasso import Validar # Importa la clase Validar del módulo ValidarRepasso / Imports Validar class from ValidarRepasso module
4
5  class Principal(): # Define la clase principal / Defines main class
6      def __init__(self): # Constructor de la clase / Class constructor
7          self.ventana = Tk() # Crea la ventana principal / Creates main window
8          self.ventana.geometry("400x200") # Define el tamaño de la ventana / Defines window size
9          self.lista = [] # Lista para almacenar datos / List to store data
10         self.valid = Validar() # Crea instancia de la clase Validar / Creates instance of Validar class
11
12     def inicio(self): # Método para crear la interfaz / Method to create interface
13         label(self.ventana, text="Programa de python con TKInter").place(x=100, y=20) # Etiqueta título con posición / Title label with position
14         Label(self.ventana, text="Escribe un dato").place(x=50, y=50) # Etiqueta instrucción / Instruction label
15         self.dato = Entry(self.ventana) # Campo de entrada para datos / Entry field for data
16         self.dato.place(x=150, y=50, width=150) # Posición y ancho del campo / Field position and width
17         Button(self.ventana, text="Validar", command=self.ValidarDatos).place(x=150, y=90, width=150) # Botón validar / Validate button
18         self.mostrar = Label(self.ventana, text="ejemplo") # Etiqueta para mostrar lista / Label to show list
19         self.mostrar.place(x=20, y=150) # Posición etiqueta mostrar / Show label position
20         self.ventana.mainloop() # Inicia el bucle principal / Starts main loop
21
22     def ValidarDatos(self): # Método para validar datos ingresados / Method to validate entered data
23         val = self.dato.get() # Obtiene el texto del campo de entrada / Gets text from entry field
24         if val != "": # Verifica que no esté vacío / Checks if not empty
25             self.Revisar(val) # Llama al método Revisar / Calls Revisar method
26             self.lista.append(val) # Agrega el valor a la lista / Adds value to list
27             self.dato.delete(0,END) # Limpia el campo de entrada / Clears entry field
28             self.mostrar.config(text=f'{self.lista}') # Actualiza la etiqueta con la lista / Updates label with list
29             respuesta = self.valid.ValidarConString(val) # Llama método de validación / Calls validation method
30             messagebox.showinfo("Validar Datos", f'El dato es {respuesta}') # Muestra resultado / Shows result
31         else:
32             messagebox.showerror("Error", "Caja de texto esta vacía") # Muestra error si está vacío / Shows error if empty
33
34     def Revisar(self, v): # Método auxiliar para revisar datos / Auxiliary method to check data
35         print(v) # Imprime el valor en consola / Prints value to console
36
37     if __name__ == '__main__':
38         app = Principal() # Crea instancia de la clase Principal / Creates instance of Principal class
39         app.inicio() # Llama al método inicio / Calls inicio method

```

Código programa 23



Código ejecución 23





Repaso2: Otro archivo de repaso, con más ejemplos o variaciones del uso de ciclos y listas.

```

1  from tkinter import *
2  from tkinter import messagebox
3  from ValidacionRepaso2 import Validacion
4
5  class Ventana():
6      def __init__(self):
7          self.ven = Tk()
8          self.ven.title('Repaso 2')
9          self.ven.geometry('600x250')
10         self.valida = Validacion()
11         self.lista = []
12         self.b1 = False
13
14     def Inicio(self):
15         Label(self.ven, text='Minusculas').place(x=10, y=10)
16         self.c1 = Entry(self.ven)
17         self.c1.place(x=10, y=35)
18
19         Label(self.ven, text='Mayusculas').place(x=150, y=10)
20         self.c2 = Entry(self.ven)
21         self.c2.place(x=150, y=35)
22
23         Label(self.ven, text='Numeros').place(x=290, y=10)
24         self.c3 = Entry(self.ven)
25         self.c3.place(x=290, y=35)
26
27         Label(self.ven, text='Datos ingresados en la lista').place(x=140,y=100)
28         self.lis = Label(self.ven, text=f'(len(self.lista))')
29         self.lis.place(x=200,y=130)
30
31         Button(self.ven, text='Validacion', command=self.validacion).place(x=110, y=70)
32         Button(self.ven, text='Aregar', command=self.agregar).place(x=255, y=70)
33
34         self.lisview = Listbox(self.ven, height=10, width=15, bg='grey', activestyle='dotbox', fg='black')#Hace de forma visual la lista
35         self.lisview.place(x=450,y=35)
36
37         self.ven.mainloop()
38
39     def validacion(self):
40         val1 = self.c1.get()
41         val2 = self.c2.get()
42         val3 = self.c3.get()
43
44         if val1 != "" and val2 != "" and val3 != "":
45             if self.valida.ValidarMinus(val1):
46                 if self.valida.ValidarMayus(val2):
47                     if self.valida.ValidarNumeros(val3):
48                         messagebox.showinfo("Validación", "Todas las cajas son correctas")
49                         self.c1.delete(0, END)
50                         self.c2.delete(0, END)
51                         self.c3.delete(0, END)
52                         self.cadena = val1+val2+val3
53                         print(self.cadena)
54                         self.b1 = True
55
56                     else:
57                         messagebox.showerror("Error", "Casilla 3 debe tener solo números")
58                         self.c3.delete(0, END)
59
59                 else:
60                     messagebox.showerror("Error", "Casilla 2 debe tener solo mayúsculas")
61                     self.c2.delete(0, END)
62
62             else:
63                 messagebox.showerror("Error", "Casilla 1 debe tener solo minúsculas")
64             else:
65                 messagebox.showerror("Error", "Todas las cajas deben estar llenas")
66
67     def agregar(self):
68         if self.b1:
69             self.lista.append(self.cadena)
70             self.lisview.insert(self.lisview.size()+1,self.cadena) #Agrega el elemento a la lista
71             self.lis.config(text=f'{len(self.lista)}') #Esto actualiza el contador de la lista
72             self.b1 = False
73
74
75 if __name__ == '__main__':
76     app = Ventana()
77     app.Inicio()
78

```

Código programa 24





Repaso 2

Minusculas	Mayusculas	Numeros
diego	PC	22

Validacion **Agregar**

Datos ingresados en la lista

0

Validación

i Todas las cajas son correctas

Aceptar

Código ejecución 24

Repaso 2

Minusculas	Mayusculas	Numeros
		1

Validacion **Agregar**

Datos ingresados en la lista

1

diegoPC22

Código ejecución 24





Tarea1.1: Versión alternativa o mejorada de la tarea anterior, también con bucles o estructuras de control.

```

1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
3
4  class Ventana(): # Define la clase Ventana / Defines Ventana class
5      def __init__(self): # Constructor de la clase / Class constructor
6          self.ven = Tk() # Crea la ventana principal / Creates main window
7          self.ven.title('Organizador de 3 calificaciones') # Establece el título / Sets the title
8          self.ven.geometry('700x350') # Define el tamaño de la ventana / Defines window size
9          self.inicio() # Llama al método inicio / Calls inicio method
10
11     def inicio(self): # Método para crear la interfaz / Method to create interface
12         label1 = Label(self.ven, text='¿Cuál es tu nombre?').pack() # Etiqueta nombre y empaqueta / Name label and packs
13         self.nom = Entry(self.ven) # Campo entrada nombre / Name entry field
14         self.nom.pack(pady=4) # Empaque con espacio / Packs with spacing
15         label1 = Label(self.ven, text='Ingresa una calificación').pack() # Etiqueta calificación 1 / Grade 1 label
16         self.c1 = Entry(self.ven) # Campo entrada calificación 1 / Grade 1 entry field
17         self.c1.pack(pady=4) # Empaque con espacio / Packs with spacing
18         label2 = Label(self.ven, text='Ingresa otra calificación').pack(pady=4) # Etiqueta calificación 2 / Grade 2 label
19         self.c2 = Entry(self.ven) # Campo entrada calificación 2 / Grade 2 entry field
20         self.c2.pack(pady=4) # Empaque con espacio / Packs with spacing
21         label3 = Label(self.ven, text='Ingresa otra calificación').pack(pady=4) # Etiqueta calificación 3 / Grade 3 label
22         self.c3 = Entry(self.ven) # Campo entrada calificación 3 / Grade 3 entry field
23         self.c3.pack(pady=4) # Empaque con espacio / Packs with spacing
24         boton = Button(self.ven, text='Ordenar', command=self.ordenar).pack(pady=4) # Botón ordenar / Sort button
25         self.ven.mainloop() # Inicia el bucle principal / Starts main loop
26
27     def ordenar(self): # Método para ordenar calificaciones / Method to sort grades
28         try:
29             self.listal = [] # Lista temporal para datos del estudiante / Temporary list for student data
30             nom = self.nom.get() # Obtiene nombre del estudiante / Gets student name
31             c1 = int(self.c1.get()) # Obtiene y convierte calificación 1 / Gets and converts grade 1
32             c2 = int(self.c2.get()) # Obtiene y convierte calificación 2 / Gets and converts grade 2
33             c3 = int(self.c3.get()) # Obtiene y convierte calificación 3 / Gets and converts grade 3
34             if(c1 > c2): # Compara calificación 1 y 2 / Compares grade 1 and 2
35                 if(c1 > c3): # Verifica si calificación 1 es la mayor / Checks if grade 1 is maximum
36                     print(f'El mayor es {c1}') # Imprime mayor / Prints maximum
37                     self.listal.append(nom) # Agrega nombre a lista / Adds name to list
38                     self.listal.append(c1) # Agrega calificación mayor / Adds maximum grade
39             if(c2 > c3): # Compara las dos calificaciones restantes / Compares remaining two grades
40                 print(f'El del medio es {c2}') # Imprime medio / Prints middle
41                 print(f'El menor es {c3}') # Imprime menor / Prints minimum
42                 self.listal.append(c2) # Agrega calificación media / Adds middle grade
43                 self.listal.append(c3) # Agrega calificación menor / Adds minimum grade
44                 lista2.append(self.listal) # Agrega a lista global / Adds to global list
45                 print(lista2) # Imprime lista global / Prints global list
46             else:
47                 print(f'El mayor es {c3}') # Imprime mayor / Prints maximum
48                 print(f'El menor es {c2}') # Imprime menor / Prints minimum
49                 self.listal.append(c3) # Agrega calificación media / Adds middle grade
50                 self.listal.append(c2) # Agrega calificación menor / Adds minimum grade
51                 lista2.append(self.listal) # Agrega a lista global / Adds to global list
52                 print(lista2) # Imprime lista global / Prints global list
53             else:
54                 print(f'El mayor es {c3}') # Imprime mayor / Prints maximum
55                 print(f'El medio es {c1}') # Imprime medio / Prints middle
56                 print(f'El menor es {c2}') # Imprime menor / Prints minimum
57                 self.listal.append(nom) # Agrega nombre a lista / Adds name to list
58                 self.listal.append(c3) # Agrega calificación mayor / Adds maximum grade
59                 self.listal.append(c1) # Agrega calificación media / Adds middle grade
60                 self.listal.append(c2) # Agrega calificación menor / Adds minimum grade
61                 lista2.append(self.listal) # Agrega a lista global / Adds to global list
62                 print(lista2) # Imprime lista global / Prints global list
63             else:
64                 if(c2 > c3): # Verifica si calificación 2 es la mayor / Checks if grade 2 is maximum
65                     print(f'El mayor es {c2}') # Imprime mayor / Prints maximum
66                     self.listal.append(nom) # Agrega nombre a lista / Adds name to list
67                     self.listal.append(c2) # Agrega calificación mayor / Adds maximum grade
68             if(c1 > c3): # Compara las dos calificaciones restantes / Compares remaining two grades
69                 print(f'El del medio es {c1}') # Imprime medio / Prints middle
70                 print(f'El menor es {c3}') # Imprime menor / Prints minimum
71                 self.listal.append(c1) # Agrega calificación media / Adds middle grade
72                 self.listal.append(c3) # Agrega calificación menor / Adds minimum grade
73                 lista2.append(self.listal) # Agrega a lista global / Adds to global list
74                 print(lista2) # Imprime lista global / Prints global list
75             else:
76                 print(f'El mayor es {c3}') # Imprime mayor / Prints maximum
77                 print(f'El medio es {c2}') # Imprime medio / Prints middle
78                 print(f'El menor es {c1}') # Imprime menor / Prints minimum
79                 self.listal.append(nom) # Agrega nombre a lista / Adds name to list
80                 self.listal.append(c3) # Agrega calificación mayor / Adds maximum grade
81                 self.listal.append(c2) # Agrega calificación media / Adds middle grade
82                 self.listal.append(c1) # Agrega calificación menor / Adds minimum grade
83                 lista2.append(self.listal) # Agrega a lista global / Adds to global list
84                 print(lista2) # Imprime lista global / Prints global list
85             except ValueError: # Captura error de conversión / Catches conversion error
86                 messagebox.showerror('Error', 'Introduce valores') # Muestra mensaje de error / Shows error message
87
88             lista2 = [] # Lista global para almacenar todos los estudiantes / Global list to store all students
89             if __name__ == '__main__':
90                 app = Ventana() # Crea instancia de la clase Ventana / Creates instance of Ventana class

```

Código programa 25





Organizador de 3 calificaciones

Cual es tu nombre

Ingresa una calificacion

Ingresa otra calificacion

Ingresa otra calificacion

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python> /Diego Plascencia/OneDrive/Escritorio/Practicas de Python> diegoPC22
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python> /Diego Plascencia/OneDrive/Escritorio/Practicas de Python> es mayor 100
El del medio es 78
El menor es 45
[[ 'Diego', 100, 78, 45]]

```

> OUTLINE

✓ TIMELINE Tarea1.1.py

- File Saved 11 hrs
- Segunda sub... 1 day
- File Saved
- File Saved 1 wk

Código ejecución 25





Tarea1: Ejercicio práctico que emplea ciclos o menús para resolver tareas específicas; probablemente parte del segundo parcial.

```

 1  from tkinter import * # Importa todos los módulos de tkinter / Imports all tkinter modules
 2  from tkinter import messagebox # Importa el módulo messagebox / Imports messagebox module
 3
 4  class Ventana(): # Define la clase Ventana / Defines Ventana class
 5      def __init__(self): # Constructor de la clase / Class constructor
 6          self.ven = Tk() # Crea la ventana principal / Creates main window
 7          self.ven.title('Organizador de 3 calificaciones') # Establece el título / Sets the title
 8          self.ven.geometry('600x300') # Define el tamaño de la ventana / Defines window size
 9          self.inicio() # Llama al método inicio / Calls inicio method
10
11     def inicio(self): # Método para crear la interfaz / Method to create interface
12         self.lis = [] # Inicializa lista para datos del estudiante / Initializes list for student data
13         labeln = Label(self.ven, text='¿Cuál es tu nombre').pack() # Etiqueta nombre y empaqueta / Name label and packs
14         self.nom = Entry(self.ven) # Campo entrada nombre / Name entry field
15         self.nom.pack(pady=4) # Empaqueuta con espacio / Packs with spacing
16         label1 = Label(self.ven, text='Ingresa una calificación').pack() # Etiqueta calificación 1 / Grade 1 label
17         self.c1 = Entry(self.ven) # Campo entrada calificación 1 / Grade 1 entry field
18         self.c1.pack(pady=4) # Empaqueuta con espacio / Packs with spacing
19         label2 = Label(self.ven, text='Ingresa otra calificación').pack(pady=4) # Etiqueta calificación 2 / Grade 2 label
20         self.c2 = Entry(self.ven) # Campo entrada calificación 2 / Grade 2 entry field
21         self.c2.pack(pady=4) # Empaqueuta con espacio / Packs with spacing
22         label3 = Label(self.ven, text='Ingresa otra calificación').pack(pady=4) # Etiqueta calificación 3 / Grade 3 label
23         self.c3 = Entry(self.ven) # Campo entrada calificación 3 / Grade 3 entry field
24         self.c3.pack(pady=4) # Empaqueuta con espacio / Packs with spacing
25         boton = Button(self.ven, text='Ordenar', command=self.ordenar).pack(pady=4) # Botón ordenar / Sort button
26         self.ven.mainloop() # Inicia el bucle principal / Starts main loop
27
28     def ordenar(self): # Método para ordenar calificaciones / Method to sort grades
29         try:
30             nombre = self.nom.get() # Obtiene nombre del estudiante / Gets student name
31             c1 = float(self.c1.get()) # Obtiene y convierte calificación 1 / Gets and converts grade 1
32             c2 = float(self.c2.get()) # Obtiene y convierte calificación 2 / Gets and converts grade 2
33             c3 = float(self.c3.get()) # Obtiene y convierte calificación 3 / Gets and converts grade 3
34             mas = max(c1,c2,c3) # Encuentra la calificación máxima / Finds maximum grade
35             mi = min(c1,c2,c3) # Encuentra la calificación mínima / Finds minimum grade
36             if mas > c1 and mi < c1: # Verifica si c1 es la calificación media / Checks if c1 is middle grade
37                 self.lis = ([nombre, mas, c1, mi]) # Crea lista ordenada [nombre, mayor, medio, menor] / Creates sorted list [name, max, middle, min]
38                 messagebox.showinfo('Correcto', 'Datos guardados con éxito') # Muestra mensaje de éxito / Shows success message
39             elif mas > c2 and mi < c2: # Verifica si c2 es la calificación media / Checks if c2 is middle grade
40                 self.lis = ([nombre, mas, c2, mi]) # Crea lista ordenada / Creates sorted list
41                 messagebox.showinfo('Correcto', 'Datos guardados con éxito') # Muestra mensaje de éxito / Shows success message
42             elif mas > c3 and mi < c3: # Verifica si c3 es la calificación media / Checks if c3 is middle grade
43                 self.lis = ([nombre, mas, c3, mi]) # Crea lista ordenada / Creates sorted list
44                 messagebox.showinfo('Correcto', 'Datos guardados con éxito') # Muestra mensaje de éxito / Shows success message
45             lista.append(self.lis) # Agrega a la lista global / Adds to global list
46             print(lista) # Imprime lista global en consola / Prints global list to console
47         except ValueError: # Captura error de conversión / Catches conversion error
48             messagebox.showerror('Error', 'Introduce valores') # Muestra mensaje de error / Shows error message
49
50     lista = [] # Lista global para almacenar todos los estudiantes / Global list to store all students
51     if __name__ == '__main__':
52         app = Ventana() # Crea instancia de la clase Ventana / Creates instance of Ventana class

```

Código programa 26





Organizador de 3 calificaciones

Cual es tu nombre

Ingrera una calificacion

Ingrera otra calificacion

Ingrera otra calificacion

.py
... U
so.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Diego Plascencia\OneDrive\Escritorio\Practicas de Python\practicaspar
/ /Diego Plascencia/OneDrive/Escritorio/Practicas de Python/practicaspar
[['diego', 34.0, 12.0, 7.0]]
```

Código ejecución 26





Validaciones: Archivo de apoyo que contiene **funciones para validar datos** (por ejemplo, verificar si una entrada es numérica o si cumple ciertas condiciones).

```
1 class validacion(): # Define la clase validacion / Defines validation class
2     def __init__(self): # Constructor de la clase / Class constructor
3         self.suma = 0 # Inicializa variable para suma / Initializes variable for sum
4         self.promedio = 0.0 # Inicializa variable para promedio / Initializes variable for average
5
6     def ValidarNumeros(self, valor): # Método para validar si es número / Method to validate if number
7         if valor.isdigit(): # Verifica si el string contiene solo dígitos / Checks if string contains only digits
8             return True # Retorna verdadero si es número / Returns true if number
9         else:
10            return False # Retorna falso si no es número / Returns false if not number
11
12    def Promedio(self, lista): # Método para calcular promedio / Method to calculate average
13        for i in lista: # Itera sobre cada elemento de la lista / Iterates over each list element
14            self.suma += i # Suma el elemento actual al total / Adds current element to total
15        self.promedio = self.suma / len(lista) # Calcula el promedio (suma/cantidad) / Calculates average (sum/count)
16        return self.promedio # Retorna el valor del promedio / Returns average value
```

Código programa 27





ValidacionReposo: Versión de validación asociada al archivo Repaso2.py, usada para comprobar la entrada de datos del usuario.

```
● ● ●  
1  class Validacion():  
2      def __init__(self):  
3          pass  
4  
5      def ValidarMinus(self, valor):  
6          if len(valor) == 0:  
7              return True  
8          if ord(valor[0]) < 97 or ord(valor[0]) > 122:  
9              return False  
10         return self.ValidarMinus(valor[1:])  
11  
12     def ValidarMayus(self, valor):  
13         if len(valor) == 0:  
14             return True  
15         if ord(valor[0]) < 65 or ord(valor[0]) > 90:  
16             return False  
17         return self.ValidarMayus(valor[1:])  
18  
19  
20     def ValidarNumeros(self, valor):  
21         if len(valor) == 0:  
22             return True  
23         if ord(valor[0]) < 48 or ord(valor[0]) > 57:  
24             return False  
25         return self.ValidarNumeros(valor[1:])  
26  
27 if __name__ == '__main__':  
28     app = Validacion()  
29  
30
```

Código programa 28





ValidarRepaso: Otro módulo de validación para los ejercicios de repaso; probablemente incluye funciones que evitan errores de entrada.

```
● ● ●  
1  class Validacion():  
2      def __init__(self):  
3          pass  
4  
5      def ValidarMinus(self, valor):  
6          if len(valor) == 0:  
7              return True  
8          if ord(valor[0]) < 97 or ord(valor[0]) > 122:  
9              return False  
10         return self.ValidarMinus(valor[1:])  
11  
12     def ValidarMayus(self, valor):  
13         if len(valor) == 0:  
14             return True  
15         if ord(valor[0]) < 65 or ord(valor[0]) > 90:  
16             return False  
17         return self.ValidarMayus(valor[1:])  
18  
19  
20     def ValidarNumeros(self, valor):  
21         if len(valor) == 0:  
22             return True  
23         if ord(valor[0]) < 48 or ord(valor[0]) > 57:  
24             return False  
25         return self.ValidarNumeros(valor[1:])  
26  
27 if __name__ == '__main__':  
28     app = Validacion()  
29  
30
```

Código programa 29





Tercera unidad, Programa1- Ordenamiento de números, y forma de eliminar en pilas y colas, el primero que entra es el ultimo que sale y el ultimo que entra es el primero que sale.

```
from tkinter import *
Click to collapse the range. import messagebox
from tkinter import ttk
import random
from Validaciones import validar

class Ventana():
    def __init__(self):
        self.val = validar()
        self.ven = Tk()
        self.ven.title('Programa 4')
        ancho = 500
        alto = 300
        ventana_alto = self.ven.winfo_screenmmwidth()
        ventana_ancho = self.ven.winfo_screenmmwidth()
        x = (ventana_alto // 2) - (ancho // 2)
        y = (ventana_ancho // 2) - (alto // 2)
        self.ven.geometry(f'{ancho}x{alto}+{x+550}+{y+150}')
        self.con = 0
        self.bandera = False
        self.renglon = -1
        self.index = 0
```

Código programa 30

```
def Inicio(self):
    Label(self.ven, text='Nombre').place(x=10, y=10)
    self.nombre = Entry(self.ven, fg="blue")
    self.nombre.place(x=10, y=30, width=100)
    #
    Label(self.ven, text='Edad').place(x=120, y=10)
    self.edad = Entry(self.ven, fg="green")
    self.edad.place(x=120, y=30, width=100)
    #
    Label(self.ven, text='Correo').place(x=230, y=10)
    self.correo = Entry(self.ven, fg="purple")
    self.correo.place(x=230, y=30, width=100)
    #
    Button(self.ven, text='Agregar', command=self.AgregarElemento).place(x=380, y=50, width=80)
    Button(self.ven, text='Eliminar', command=self.Eliminar).place(x=380, y=90, width=80)
    Button(self.ven, text='Seleccionar', command=self.Seleccionar).place(x=380, y=130, width=80)
    #dataGrind
    columnas = ("Clave", "Nombre", "Correo", "Edad")
    self.tabla = ttk.Treeview(self.ven, columns=columnas, show='headings')
    self.tabla.place(x=10, y=100, width=350, height=190)
    for col in columnas:
        self.tabla.heading(col, text=col)
        self.tabla.column(col, anchor='center', width=30)
    scrollx = ttk.Scrollbar(self.ven, orient='horizontal', command=self.tabla.xview)
    scrollx.place(x=360, y=90, height=200)
    scrollx.place(x=10, y=280, width=350)
    self.ven.mainloop()
```

Código programa 30





```
def Seleccionar(self):
    self.renglon = self.tabla.selection()
    if not self.renglon:
        messagebox.showerror('Error', 'Elije una fila')
    else:
        valores = self.tabla.item(self.renglon, "values")
        print(valores)
        self.index = valores[0]
        self.index = self.index[:len(self.index)-2]
        print(self.index)
        self.nombre.insert(0, valores[1])
        self.edad.insert(0, valores[3])
        self.correo.insert(0, valores[2])
        self.bandera = True
```

Código programa 30

```
def AgregarElemento(self):
    messagebox.showerror('Error', 'Faltan datos')
else:
    nombre = self.nombre.get()
    edad = self.edad.get()
    correo = self.correo.get()
    if self.bandera == False:
        self.con += 1
        clave = str(self.con)+str(random.randint(1,100))+self.nombre.get()[0:2].upper()
        self.tabla.insert("", "end", values=(clave, nombre, edad, correo))
        self.nombre.delete(0,END)
        self.edad.delete(0,END)
        self.correo.delete(0,END)
        self.bandera = True
        self.renglon = -1
        messagebox.showinfo('Correcto', 'Datos actualizados')

def Eliminar(self):
    self.renglon = self.tabla.selection()
    if not self.renglon:
        messagebox.showerror('Error', 'Elije una fila')
    else:
        messagebox.showinfo('Error', 'Fila eliminada')
        self.tabla.delete(self.renglon)

if __name__ == '__main__':
    app = Ventana()
    app.Inicio()
```

Código programa 30





Programa 1

Validar Eliminar Ordenar

Numero de elementos: 1

Pilas Colas

Ordenar burbuja

Ordenar selección

23

Código ejecución 27





Programa 2-Calculadora de RFC

```

1  from tkinter import *
2  from tkinter import messagebox
3  from Validaciones2 import validar
4  class Ventana():
5      def __init__(self):
6          self.ven = Tk()
7          self.ven.title('Programa 2')
8          self.ven.geometry('650x300')
9          self.val = validar()
10         self.lista = []
11     def Inicio(self):
12         Label(self.ven,text='Nombre').place(x=10,y=10)
13         self.nombre = Entry(self.ven)
14         self.nombre.place(x=10,y=40)
15         Label(self.ven,text='Apellido paterno').place(x=165,y=10)
16         self.paterno = Entry(self.ven)
17         self.paterno.place(x=165,y=40)
18         Label(self.ven,text='Apellido materno').place(x=320,y=10)
19         self.materno = Entry(self.ven)
20         self.materno.place(x=320,y=40)
21         # -----
22         Label(self.ven,text='FECHA DE NACIMIENTO').place(x=175,y=80)
23         Label(self.ven,text='Dia').place(x=10,y=120)
24         self.dia = Entry(self.ven)
25         self.dia.place(x=10,y=150)
26         Label(self.ven,text='Mes').place(x=165,y=120)
27         self.mes = Entry(self.ven)
28         self.mes.place(x=165,y=150)
29         Label(self.ven,text='Año').place(x=320,y=120)
30         self.ano = Entry(self.ven)
31         self.ano.place(x=320,y=150)
32         # -----
33         self.modo = StringVar(value='Pilas')
34         Button(self.ven,text='Calcular RFC',command=self.Calcular).place(x=10,y=180)
35         Button(self.ven,text='Eliminar',command=self.Eliminar).place(x=320,y=180)
36         Radiobutton(self.ven,text='Pilas',variable=self.modo, value='Pilas').place(x=165, y=180)
37         Radiobutton(self.ven,text='Colas',variable=self.modo, value='Colas').place(x=235, y=180)
38         self.listview = Listbox(self.ven, height=10, width=15, bg='grey', activestyle="dotbox", fg="Black")
39         self.listview.place(x=480, y=10)
40         self.ven.mainloop()

```

Código programa 31





```

  ● ○ ●
1  def Calcular(self):
2      a = False
3      b = False
4      c = False
5      d = False
6      e = False
7      f = False
8      nombre = self.nombre.get()
9      paterno = self.paterno.get()
10     materno = self.materno.get()
11     dia = self.dia.get()
12     mes = self.mes.get()
13     anio = self.anio.get()
14     if nombre != "" and paterno != "" and materno != "" and dia != "" and mes != "" and anio != "":
15         if self.val.ValidarLetra(nombre):
16             a = True
17         else:
18             messagebox.showerror('Error','Nombre incorrecto')
19             self.nombre.delete(0,END)
20         if self.val.ValidarLetra(paterno):
21             b = True
22         else:
23             messagebox.showerror('Error','Apellido paterno incorrecto')
24             self.paterno.delete(0,END)
25         if self.val.ValidarLetra(materno):
26             c = True
27         else:
28             messagebox.showerror('Error','Apellido materno incorrecto')
29             self.materno.delete(0,END)
30         if self.val.ValidarNumeros(dia):
31             if int(dia) < 1 or int(dia) > 31:
32                 messagebox.showerror('Error','Dia inválido')
33                 self.dia.delete(0,END)
34             else:
35                 d = True
36         else:
37             messagebox.showerror('Error','Dia inválido')
38             self.mes.delete(0,END)
39         if self.val.ValidarNumeros(mes):
40             if int(mes) < 1 or int(mes) > 12:
41                 messagebox.showerror('Error','Mes inválido')
42                 self.mes.delete(0,END)
43             else:
44                 e = True
45         else:
46             messagebox.showerror('Error','Mes inválido')
47             self.anio.delete(0,END)
48         if self.val.ValidarNumeros(anio):
49             if len(anio) > 4:
50                 messagebox.showerror('Error','Año inválido')
51                 self.anio.delete(0,END)
52             else:
53                 f = True
54         else:
55             messagebox.showerror('Error','Año inválido')
56             self.anio.delete(0,END)
57     if a == True and b == True and c == True and d == True and e == True and f == True:
58         # print('Todo correcto')
59         self.rfc = paterno[0:2].upper()+materno[0].upper()+nombre[0].upper()+anio[2:]+mes.zfill(2)+dia.zfill(2)
60         self.listview.insert(self.listview.size(),self.rfc)
61         self.nombre.delete(0,END)
62         self.paterno.delete(0,END)
63         self.materno.delete(0,END)
64         self.dia.delete(0,END)
65         self.mes.delete(0,END)
66         self.anio.delete(0,END)
67     else:
68         messagebox.showerror('Error', "Cajas de texto vacías")
69         self.nombre.delete(0,END)
70         self.paterno.delete(0,END)
71         self.materno.delete(0,END)
72         self.dia.delete(0,END)
73         self.mes.delete(0,END)
74         self.anio.delete(0,END)
75     def Eliminar(self):

```

Código programa 32





```
1  def Eliminar(self):
2      if self.listview.size() <= 0:
3          messagebox.showerror('Error','La lista esta vacia')
4          return
5      if self.modo.get() == 'Pilas':
6          #El ultimo que entra es el primero que sale
7          self.listview.delete(self.listview.size()-1)
8      else:
9          #El primero que entra es el priemro que sale
10         self.listview.delete(0)
11 if __name__ == '__main__':
12     app = Ventana()
13     app.Inicio()
```

Código programa 32





Programa 2

Nombre	Apellido paterno	Apellido materno	PLCD061016
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

FECHA DE NACIMIENTO

Dia	Mes	Año
<input type="text"/>	<input type="text"/>	<input type="text"/>

[Calcular RFC](#) Pilas Colas [Eliminar](#)

Código ejecución 28





Programa 3-Se pide lo que es nombre, numero de teléfono, correo. Agregándose a una lista y concatenándose para poderse mostrar.

```

1  from tkinter import *           # Importa todas las clases y funciones de Tkinter / Imports all Tkinter classes and functions
2  from tkinter import messagebox   # Importa la función de mensajes emergentes / Imports the messagebox function
3  from Validaciones import validar # Importa la clase 'validar' del archivo Validaciones.py / Imports the 'validar' class from Validaciones.py
4
5  class Ventana():
6      def __init__(self):
7          self.val = validar()         # Crea un objeto de la clase validar / Creates an object from the validar class
8          self.ven = Tk()              # Crea la ventana principal de la aplicación / Creates the main application window
9          self.ven.title('Programa 3') # Asigna el título de la ventana / Sets the window title
10         ancho = 350                # Define el ancho de la ventana / Defines window width
11         alto = 250                 # Define el alto de la ventana / Defines window height
12         ventana_alto = self.ven.winfo_screenmmwidth() # Obtiene el alto de la pantalla en milímetros / Gets screen height in millimeters
13         ventana_ancho = self.ven.winfo_screenmmwidth() # Obtiene el ancho de la pantalla en milímetros / Gets screen width in millimeters
14         x = (ventana_alto // 2) - (ancho // 2) # Calcula la posición horizontal centrada / Calculates centered horizontal position
15         y = (ventana_ancho // 2) - (alto // 2) # Calcula la posición vertical centrada / Calculates centered vertical position
16         self.ven.geometry(f'{ancho}x{alto}+{x+550}+{y+150}') # Define el tamaño y posición de la ventana / Sets window size and position
17
18      # ----- MÉTODOS PARA PLACEHOLDERS -----
19      def quitar_placeholder1(self, event):
20          # Si el texto actual es igual al placeholder, se borra el texto y se cambia el color /
21          # If the current text equals the placeholder, delete it and change text color
22          if self.nombre.get() == self.placeholder1:
23              self.nombre.delete(0, END)
24              self.nombre.config(fg="black")
25
26      def poner_placeholder1(self, event):
27          # Si el campo está vacío, se vuelve a poner el placeholder /
28          # If the field is empty, reinsert the placeholder
29          if self.nombre.get() == "":
30              self.nombre.insert(0, self.placeholder1)
31              self.nombre.config(fg="gray")
32
33      def quitar_placeholder2(self, event):
34          # Borra el texto del segundo campo si es igual al placeholder /
35          # Clears text if it matches placeholder (for second entry)
36          if self.telefono.get() == self.placeholder2:
37              self.telefono.delete(0, END)
38              self.telefono.config(fg="black")
39
40      def poner_placeholder2(self, event):
41          # Restaura el placeholder si el campo está vacío /
42          # Restores placeholder if field is empty
43          if self.telefono.get() == "":
44              self.telefono.insert(0, self.placeholder2)
45              self.telefono.config(fg="gray")
46
47      def quitar_placeholder3(self, event):
48          # Borra el texto del tercer campo si es igual al placeholder /
49          # Clears text if it matches placeholder (for third entry)
50          if self.domicilio.get() == self.placeholder3:
51              self.domicilio.delete(0, END)
52              self.domicilio.config(fg="black")
53
54      def poner_placeholder3(self, event):
55          # Restaura el placeholder si el campo está vacío /
56          # Restores placeholder if field is empty
57          if self.domicilio.get() == "":
58              self.domicilio.insert(0, self.placeholder3)
59              self.domicilio.config(fg="gray")
60
61      # ----- CONFIGURACIÓN DE LA INTERFAZ -----

```

Código programa 33





```

1 def Inicio(self):
2     # Campo de texto para el nombre / Name input field
3     self.placeholder1 = 'Nombre'
4     self.nombre = Entry(self.ven, fg="gray")           # Crea una caja de texto con texto gris / Creates an entry with gray text
5     self.nombre.insert(0, self.placeholder1)           # Inserta el texto "Nombre" como placeholder / Inserts "Nombre" as placeholder
6     self.nombre.bind("<FocusIn>", self.guitar_placeholder1) # Quita el placeholder al enfocar / Removes placeholder on focus
7     self.nombre.bind("<FocusOut>", self.poner_placeholder1) # Restaura placeholder al perder foco / Restores placeholder on focus out
8     self.nombre.bind("<Return>", self.ValidarCaja)      # Valida al presionar Enter / Validates on pressing Enter
9     self.nombre.place(x=10, y=10, width=100)           # Coloca el campo en pantalla / Places the field on screen
10
11    # Campo de texto para el teléfono / Phone input field
12    self.placeholder2 = "Telefono"
13    self.telefono = Entry(self.ven, fg="gray")          # Crea la caja de texto gris / Creates gray text entry
14    self.telefono.insert(0, self.placeholder2)           # Inserta el texto "Telefono" como placeholder / Inserts "Telefono" placeholder
15    self.telefono.bind("<FocusIn>", self.guitar_placeholder2)
16    self.telefono.bind("<FocusOut>", self.poner_placeholder2)
17    self.telefono.bind("<Return>", self.ValidarCaja)
18    self.telefono.place(x=120, y=10, width=100)
19
20    # Campo de texto para domicilio / Address input field
21    self.domicilio = Entry(self.ven, fg="gray")
22    self.placeholder3 = "Domicilio"                     # (Error tipográfico: debería ser Domicilio) / Typo: should be "Domicilio"
23    self.domicilio.insert(0, self.placeholder3)
24    self.domicilio.bind("<FocusIn>", self.guitar_placeholder3)
25    self.domicilio.bind("<FocusOut>", self.poner_placeholder3)
26    self.domicilio.bind("<Return>", self.ValidarCaja)
27    self.domicilio.place(x=230, y=10, width=100)
28
29    # Etiqueta y botones de selección de sexo / Label and radio buttons for gender
30    Label(self.ven, text='Sexo').place(x=10, y=30)        # Muestra el texto "Sexo" / Displays the word "Sexo"
31    self.modo = StringVar(value='F')                      # Variable para guardar el valor seleccionado / Variable to store selected value
32    Radiobutton(self.ven, text='F', variable=self.modo, value='F').place(x=10, y=50) # Opción femenina / Female option
33    Radiobutton(self.ven, text='M', variable=self.modo, value='M').place(x=10, y=70) # Opción masculina / Male option
34
35    # Lista para mostrar los datos agregados / Listbox to display added data
36    self.lista = listbox(self.ven, height=8, width=30, bg='grey', activestyle="dotbox", fg="Black")
37    self.lista.place(x=10, y=100)
38
39    # Botón para agregar a la lista / Button to add data to the list
40    Button(self.ven, text='Añadir', command=self.ValidarCaja, width=10).place(x=210, y=100, width=100, height=50)
41
42    # Mantiene la ventana abierta / Keeps window running
43    self.ven.mainloop()

```

Código programa 32





```

1  def ValidarCaja(self, event=0):
2      a = False # Variable para validar nombre / Flag to validate name
3      b = False # Variable para validar teléfono / Flag to validate phone number
4
5      # Verifica que los campos no estén vacíos o con placeholders / Checks that fields are not empty or placeholders
6      if (self.nombre.get() == self.placeholder1 or
7          self.telefono.get() == self.placeholder2 or
8          self.domicilio.get() == self.placeholder3 or
9          self.domicilio.get() == ""):
10         messagebox.showerror('Error', 'Faltan datos') # Muestra error si faltan datos / Shows error if fields are missing
11     else:
12         # Obtiene los valores de los campos / Gets input values
13         nombre = self.nombre.get()
14         telefono = self.telefono.get()
15         domicilio = self.domicilio.get()
16
17         # Determina el sexo según el radiobutton / Determines gender based on radio button
18         if self.modo.get() == 'F':
19             sexo = 'Femenino'
20         else:
21             sexo = 'Masculino'
22
23         # Valida que el nombre contenga solo letras / Checks that name has only letters
24         if self.val.ValidarLetra(nombre):
25             a = True
26         else:
27             self.nombre.delete(0, END)
28             messagebox.showerror('Error', 'Nombre incorrecto') # Error si contiene caracteres inválidos / Error if invalid characters
29
30         # Valida que el teléfono contenga solo números / Checks that phone number has only digits
31         if self.val.ValidarNumeros(telefono):
32             if len(telefono) == 10: # Requiere 10 dígitos / Requires 10 digits
33                 b = True
34             else:
35                 self.telefono.delete(0, END)
36                 messagebox.showerror('Error', 'Teléfono incorrecto')
37         else:
38             self.telefono.delete(0, END)
39             messagebox.showerror('Error', 'Teléfono incorrecto')
40
41         # Si ambas validaciones son correctas / If both validations pass
42         if a == True and b == True:
43             clave = nombre[0]+telefono[0]+domicilio[0] # Crea una clave con las iniciales / Creates key with initials
44             persona = clave+"-"+nombre+"-"+telefono+"-"+domicilio+"-"+sexo # Une toda la información en una cadena / Combines info into a string
45             self.lista.insert(self.lista.size()+1, persona) # Agrega la persona a la lista / Inserts person into listbox
46

```

Código programa 33





Programa 3

diego	3781183263	De los Olivos #139
-------	------------	--------------------

Sexo

F

M

d3D-diego-3781183263-De los Oliv

Agregar

Código ejecución 29





Programa 4-Tabla donde se introduce lo que es el nombre, edad, correo y la clave que se creó a base de los elementos ya mencionados y usando la librería random para agregar uno que otro valor random a la cadena.

```

1 # Importamos los módulos necesarios de Tkinter
2 # Import the necessary Tkinter modules
3 from tkinter import *
4 from tkinter import messagebox
5 from tkinter import ttk
6 import random
7 from Validaciones import validar # Importamos una clase personalizada para validaciones / Custom validation class
8
9 # Definición de la clase principal Ventana / Definition of the main class Window
10 class Ventana():
11     def __init__(self):
12         # Creamos una instancia de la clase validar / Create an instance of the validation class
13         self.val = validar()
14         # Creamos la ventana principal / Create the main window
15         self.ven = Tk()
16         self.ven.title('Programa 4') # Título de la ventana / Window title
17
18         # Dimensiones de la ventana / Window dimensions
19         ancho = 500
20         alto = 300
21         ventana_alto = self.ven.winfo_screenmmwidth() # Obtiene el ancho de pantalla en milímetros / Gets screen width in mm
22         ventana_ancho = self.ven.winfo_screenmmwidth() # (Posible error: debería ser height) / (Possible error: should be height)
23         x = (ventana_alto // 2) - (ancho // 2) # Calcula posición X centrada / Calculates centered X position
24         y = (ventana_ancho // 2) - (alto // 2) # Calcula posición Y centrada / Calculates centered Y position
25         self.ven.geometry(f'{ancho}x{alto}+{x+550}+{y+150}') # Define tamaño y posición / Set size and position
26
27         # Variables de control / Control variables
28         self.con = 0          # Contador para generar claves / Counter to generate IDs
29         self.bandera = False   # Bandera para modo edición / Flag for edit mode
30         self.renglon = -1      # Índice de fila seleccionada / Selected row index
31         self.index = 0          # Índice de clave actual / Current key index

```

Código programa 34





```

1 def Inicio(self):
2     # Etiqueta y campo de entrada para el nombre / Label and entry for name
3     Label(self.ven,text='Nombre').place(x=10,y=10)
4     self.nombre = Entry(self.ven,fg="blue")
5     self.nombre.place(x=10, y=30, width=100)
6
7     #-----
8     # Etiqueta y campo de entrada para la edad / Label and entry for age
9     Label(self.ven,text='Edad').place(x=120,y=10)
10    self.edad = Entry(self.ven,fg="green")
11    self.edad.place(x=120, y=30, width=100)
12
13    #-----
14    # Etiqueta y campo de entrada para el correo / Label and entry for email
15    Label(self.ven,text='Correo').place(x=230,y=10)
16    self.correo = Entry(self.ven,fg="purple")
17    self.correo.place(x=230, y=30, width=100)
18
19    #-----
20    # Botones con sus respectivas funciones / Buttons with their functions
21    Button(self.ven,text='Agregar',command=self.AgregarElemento).place(x=380,y=50,width=80)
22    Button(self.ven,text='Eliminar',command=self.Eliminar).place(x=380,y=90,width=80)
23    Button(self.ven,text='Seleccionar',command=self.Seleccionar).place(x=380,y=130,width=80)
24
25    #-----
26    # Creación de la tabla / Create the data table
27    columnas = ("Clave","Nombre","Corre", "Edad")
28    self.tabla = ttk.Treeview(self.ven,columns=columnas,show='headings')
29    self.tabla.place(x=10,y=100,width=350,height=190)
30
31    # Configuración de los encabezados / Set up column headers
32    for col in columnas:
33        self.tabla.heading(col, text=col)
34        self.tabla.column(col, anchor='center', width=30)
35
36    # Barras de desplazamiento / Scrollbars
37    scrollly = ttk.Scrollbar(self.ven,orient='vertical', command=self.tabla.yview)
38    scrollx = ttk.Scrollbar(self.ven,orient='horizontal', command=self.tabla.xview)
39    scrollly.place(x=360,y=90,height=200)
40    scrollx.place(x=10,y=280,width=350)
41
42    # Iniciar la ventana principal / Start the main window loop
43    self.ven.mainloop()

```

Código programa 34





```

1 def Seleccionar(self):
2     # Obtiene el renglón seleccionado / Get selected row
3     self.renglon = self.tabla.selection()
4     if not self.renglon:
5         # Muestra un mensaje si no se seleccionó nada / Show message if no selection
6         messagebox.showerror('Error', 'Elija una fila')
7     else:
8         # Obtiene los valores de la fila seleccionada / Get values from selected row
9         valores = self.tabla.item(self.renglon,"values")
10        print(valores)
11        # Extrae el índice de la clave / Extract ID index
12        self.index = valores[0]
13        self.index = self.index[:len(self.index)-2]
14        print(self.index)
15        # Inserta los valores en los campos de texto / Insert values into entry fields
16        self.nombre.insert(0,valores[1])
17        self.edad.insert(0,valores[3])
18        self.correo.insert(0,valores[2])
19        self.bandera = True # Activa el modo edición / Activate edit mode
20
21 def AgregarElemto(self):
22     # Verifica si hay campos vacíos / Check if any field is empty
23     if len(self.nombre.get()) == 0 or len(self.edad.get()) == 0 or len(self.correo.get()) == 0:
24         messagebox.showerror('Error', 'Faltan datos')
25     else:
26         # Obtiene los datos de las entradas / Get data from entries
27         nombre = self.nombre.get()
28         edad = self.edad.get()
29         correo = self.correo.get()
30
31         # Si no está en modo edición / If not in edit mode
32         if self.bandera == False:
33             self.con += 1 # Incrementa contador / Increase counter
34             # Genera una clave única / Generate unique key
35             clave = str(self.con)+str(random.randint(1,100))+self.nombre.get()[0:2].upper()
36             # Inserta en la tabla / Insert into table
37             self.tabla.insert("", "end",values=(clave,nombre,correo,edad))
38             # Limpia los campos / Clear entry fields
39             self.nombre.delete(0,END)
40             self.edad.delete(0,END)
41             self.correo.delete(0,END)
42         else:
43             # Si está en modo edición / If in edit mode
44             clave = self.index+self.nombre.get()[0:2].upper()
45             print('Modo edición activado')
46             # Actualiza los valores de la fila seleccionada / Update selected row values
47             self.tabla.item(self.renglon,values=(clave,nombre,correo,edad))
48             # Limpia los campos / Clear fields
49             self.nombre.delete(0,END)
50             self.edad.delete(0,END)
51             self.correo.delete(0,END)
52             # Restablece los valores / Reset values
53             self.bandera = False
54             self.renglon = -1
55             messagebox.showinfo('Correcto', 'Datos actualizados')
56

```

Código programa 34





```

1  def Eliminar(self):
2      # Obtiene el renglón seleccionado / Get selected row
3      self.renglon = self.tabla.selection()
4      if not self.renglon:
5          # Si no hay selección / If no selection
6          messagebox.showerror('Error', 'Elige una fila')
7      else:
8          # Muestra mensaje de eliminación y borra la fila / Show message and delete row
9          messagebox.showinfo('Error', 'Fila eliminada')
10         self.tabla.delete(self.renglon)
11

```

Código programa 34

Programa 4

Nombre	Edad	Correo
<input type="text"/>	<input type="text"/>	<input type="text"/>

Clave	Nombre	Corre	Edad
1100DI	Diego	diego@diego.c	19

Código ejecución 30





Unidad 4

Programa 1: El programa es una aplicación gráfica hecha en Tkinter que permite al usuario escribir dos números en una primera ventana. Al enviarlos, se abre una segunda ventana donde el usuario puede ver un mensaje y realizar la suma de los dos números ingresados, mostrando el resultado mediante un cuadro de diálogo. También incluye un botón para regresar a la ventana principal y volver a ingresar nuevos valores.

```

1  def Enviar(self):
2      try:
3          n1 = int(self.n1.get())
4          n2 = int(self.n2.get())
5          self.n1.delete(0,END)
6          self.n2.delete(0,END)
7          self.ven.withdraw()
8          otra = Toplevel(self.ven)
9          Ventana(otra,self.ven,n1,n2)
10         except ValueError:
11             self.n1.delete(0,END)
12             self.n2.delete(0,END)
13             messagebox.showerror('Error','Algun dato no es numero')
14
15     def Cerrar(self):
16         self.ven.destroy()
17
18 class Ventanadatos():
19     def __init__(self,master,ven,a,b):
20         self.n1 = a
21         self.n2 = b
22         self.dos = master
23         self.dos.title('Programa 1')
24         ancho = 250
25         alto = 200
26         ventana_alto = self.dos.winfo_screenmmwidth()
27         ventana_ancho = self.dos.winfo_screenmmwidth()
28         x = (ventana_alto // 2) - (ancho // 2)
29         y = (ventana_ancho // 2) - (alto // 2)
30         self.dos.geometry(f'{ancho}x{alto}+{x+550}+{y+150}')
31         Label(self.dos,text='Hola mundo').place(x=50,y=50)
32         Button(self.dos,text='Regresar',command=self.Regresar).place(x=50,y=100)
33         self.ven = ven
34         Button(self.dos,text='Sumar',command=self.sumar).place(x=150,y=100)
35
36     def sumar(self):
37         messagebox.showinfo('La suma',f'La suma de {self.n1} mas {self.n2} es \n {self.n1 + self.n2}')
38     def Regresar(self):
39         self.dos.destroy()
40         self.ven.deiconify()

```

```

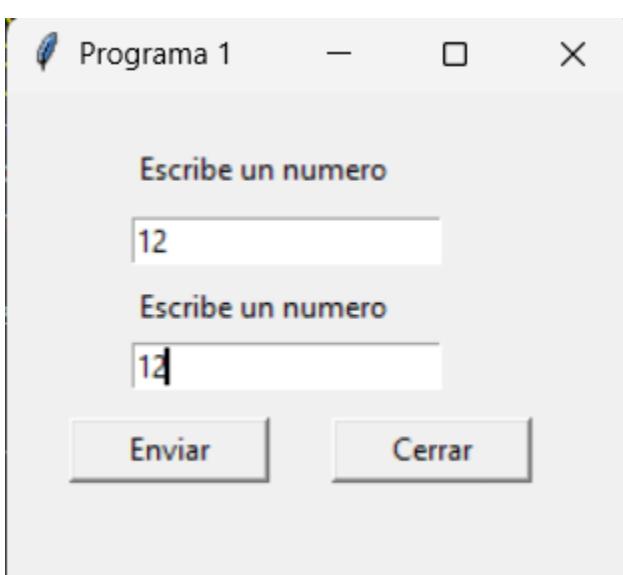
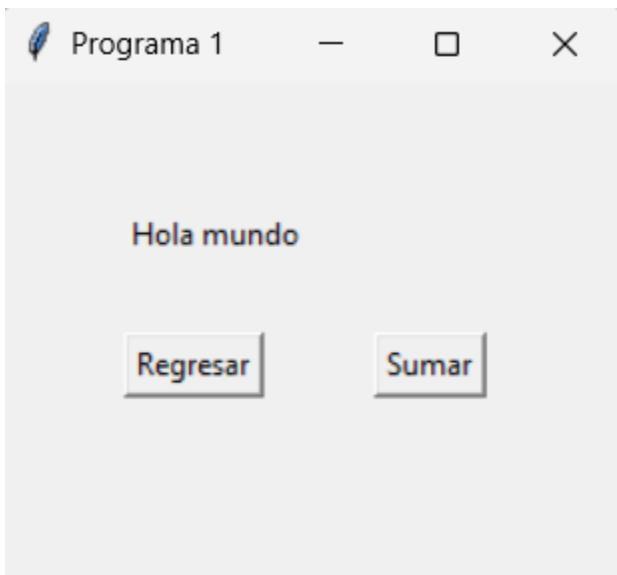
1  from tkinter import *
2  from tkinter import messagebox
3
4  class Ventana():
5      def __init__(self,master):
6          self.ven = master
7          self.ven.title('Programa 1')
8          ancho = 250
9          alto = 200
10         ventana_alto = self.ven.winfo_screenmmwidth()
11         ventana_ancho = self.ven.winfo_screenmmwidth()
12         x = (ventana_alto // 2) - (ancho // 2)
13         y = (ventana_ancho // 2) - (alto // 2)
14         self.ven.geometry(f'{ancho}x{alto}+{x+550}+{y+150}')
15
16
17     def Inicio(self):
18         Label(self.ven,text='Escribe un numero').place(x=50,y=20)
19         self.n1 = Entry(self.ven)
20         self.n1.place(x=50,y=50)
21         Label(self.ven,text='Escribe un numero').place(x=50,y=75)
22         self.n2 = Entry(self.ven)
23         self.n2.place(x=50,y=100)
24         Button(self.ven,text='Enviar',command=self.Envir,width=10).place(x=25,y=130)
25         Button(self.ven,text='Cerrar',command=self.Cerrar,width=10).place(x=130,y=130)

```





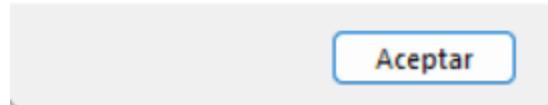
```
● ● ●  
1 if __name__ == '__main__':  
2     master = Tk()  
3     app = Ventana(master)  
4     app.Inicio()  
5     master.mainloop()
```





La suma de 12 mas 12 es
24

Aceptar





Programa 2: El programa es un sistema de login y administración de usuarios creado en Python con Tkinter y SQLite. Primero valida el acceso mediante un usuario y contraseña guardados en una base de datos ("Usuarios.db"). Si el inicio de sesión es correcto, abre una segunda ventana donde se muestra una tabla con todos los usuarios y se habilitan las opciones del menú según el rol del usuario (admin, supervisor o jefe de área). Dependiendo del rol, el usuario podrá agregar, modificar o eliminar cuentas. La tabla se actualiza dinámicamente y se evita eliminar al usuario que está actualmente en sesión.

```

1  from tkinter import *
2  from tkinter import messagebox
3  from tkinter import ttk
4  import tkinter as tk
5  import sqlite3
6
7  def crearBaseDatos():
8      con = sqlite3.connect("Usuarios.db")
9      cursor = con.cursor()
10
11     cursor.execute("""CREATE TABLE IF NOT EXISTS usuarios(
12             id INTEGER PRIMARY KEY AUTOINCREMENT,
13             usuario TEXT NOT NULL,password TEXT NOT NULL)""")
14
15     cursor.execute("SELECT * FROM usuarios WHERE usuario='admin'")
16     if not cursor.fetchone():
17         cursor.execute("INSERT INTO usuarios (usuario,password) VALUES (?,?)",("admin","12345"))
18
19     con.commit()
20     con.close()
21
22 class Ventana():
23     def __init__(self,master):
24         self.ven = master
25         self.ven.title('Programa 2')
26         ancho = 250
27         alto = 200
28         ventana_ancho = self.ven.winfo_screenwidth()
29         ventana_alto = self.ven.winfo_screenheight()
30         x = (ventana_ancho // 2) - (ancho // 2)
31         y = (ventana_alto // 2) - (alto // 2)
32         self.ven.geometry(f'{ancho}x{alto}+{x}+{y}')
33
34     def Inicio(self):
35         Label(self.ven,text='Usuario').place(x=50,y=20)
36         self.n1 = Entry(self.ven)
37         self.n1.place(x=50,y=50)
38
39         Label(self.ven,text='Password').place(x=50,y=75)
40         self.n2 = Entry(self.ven, show="*")
41         self.n2.place(x=50,y=100)
42
43         Button(self.ven,text='Validar',command=self.Enviar,width=10,fg='green').place(x=30,y=140)
44         Button(self.ven,text='Cerrar',command=self.Cerrar,width=10,fg='red').place(x=150,y=140)
45
46     def Enviar(self):
47         u = self.n1.get()
48         p = self.n2.get()
49
50         con = sqlite3.connect("Usuarios.db")
51         cursor = con.cursor()
52         cursor.execute("SELECT * FROM usuarios WHERE usuario=? and password=?", (u,p))
53         resultado = cursor.fetchone()
54         con.close()
55
56         if resultado:
57             self.n1.delete(0,END)
58             self.n2.delete(0,END)
59             self.ven.withdraw()
60             otra = Toplevel(self.ven)
61             Ventanados(otra,self.ven,u)
62         else:
63             messagebox.showerror('Error','Datos incorrectos')
64             self.n1.delete(0,END)
65             self.n2.delete(0,END)
66
67     def Cerrar(self):
68         self.ven.destroy()
69

```





```

1  class Ventanados():
2      def __init__(self,master,ven,u):
3          self.ven = ven
4          self.usuario = u
5          self.dos = master
6          self.dos.title('Programa 1')
7
8          ancho = 500
9          alto = 320
10         ventana_ancho = self.dos.winfo_screenwidth()
11         ventana_alto = self.dos.winfo_screenheight()
12         x = (ventana_ancho // 2) - (ancho // 2)
13         y = (ventana_alto // 2) - (alto // 2)
14         self.dos.geometry(f'{ancho}x{alto}+{x}+{y}')
15
16         Button(self.dos,text='Regresar',command=self.Regresar,width=10).place(x=150,y=140)
17
18         self.us = Label(self.dos,text=f'Bienvenido \n {self.usuario}')
19         self.us.place(x=400,y=1)
20
21         self.Mostrar()
22         self.mostrarTabla()
23
24         self.menus = tk.Menu(self.dos)
25         self.dos.config(menu=self.menus)
26
27         self.archivos = tk.Menu(self.menus,tearoff=0)
28         self.archivos.add_command(label='Aregar',command=self.Crearusuario)
29         self.indexAregar = self.archivos.index("end")
30         self.archivos.add_command(label='Modificar',command=self.Modificarus)
31         self.indexModificar = self.archivos.index("end")
32         self.archivos.add_command(label='Eliminar',command=self.Eliminarusr)
33         self.indexEliminar = self.archivos.index("end")
34         self.archivos.add_command(label='Salir',command=self.salir)
35         self.menus.add_cascade(label='Archivo',menu=self.archivos)
36
37         self.archivos.entryconfig(self.indexAregar,state = 'disable')
38         self.archivos.entryconfig(self.indexEliminar,state = 'disable')
39         self.archivos.entryconfig(self.indexModificar,state = 'disable')
40
41         self.roles()
42     def roles(self):
43         if self.usuario == 'admin':
44             self.archivos.entryconfig(self.indexAregar,state='normal')
45             self.archivos.entryconfig(self.indexEliminar,state='normal')
46             self.archivos.entryconfig(self.indexModificar,state='normal')
47
48         elif self.usuario == 'Supervisor' or self.usuario == 'supervisor':
49             self.archivos.entryconfig(self.indexAregar,state='normal')
50             self.archivos.entryconfig(self.indexEliminar,state='disable')
51             self.archivos.entryconfig(self.indexModificar,state='disable')
52
53         elif self.usuario == 'Jefe de area' or self.usuario == 'jefe de area':
54             self.archivos.entryconfig(self.indexAregar,state='disable')
55             self.archivos.entryconfig(self.indexEliminar,state='normal')
56             self.archivos.entryconfig(self.indexModificar,state='disable')
57
# -----

```





```

1  def Crearusuario(self):
2      if len(self.usu.get()) != 0 and len(self.pas.get()) != 0:
3          con = sqlite3.connect("usuarios.db")
4          cursor = con.cursor()
5          cursor.execute("INSERT INTO usuarios (usuario,password) VALUES (?,?)", (self.usu.get(), self.pas.get()))
6          con.commit()
7          con.close()
8          self.Actualizartabla()
9          self.borrarcaja("Usuario agregado correctamente")
10         else:
11             messagebox.showerror('Error','Faltan Datos')
12
13     def Seleccionfila(self, event):
14         try:
15             index = self.tabla.selection()[0]
16         except:
17             return
18
19         valores = self.tabla.item(index,"values")
20         self.usu.delete(0,END)
21         self.pas.delete(0,END)
22         self.usu.insert(0, valores[1])
23         self.pas.insert(0, valores[2])
24
25     def Actualizartabla(self):
26         for i in self.tabla.get_children():
27             self.tabla.delete(i)
28         self.mostrarTabla()
29
30     def Modificarus(self):
31         try:
32             index = self.tabla.selection()[0]
33         except:
34             messagebox.showerror('Error','Elige un usuario')
35             return
36
37         valores = self.tabla.item(index,"values")
38         id = valores[0]
39
40         if len(self.usu.get()) != 0 and len(self.pas.get()) != 0:
41             p = self.pas.get()
42             con = sqlite3.connect("usuarios.db")
43             cursor = con.cursor()
44             cursor.execute("UPDATE usuarios SET usuario=? , password=? WHERE id=? ", (u, p, id))
45             con.commit()
46             con.close()
47             self.Actualizartabla()
48             self.borrarcaja("Usuario modificado correctamente")
49         else:
50             messagebox.showerror('Error','Faltan datos')
51
52     def Eliminarusr(self):
53         try:
54             index = self.tabla.selection()[0]
55             valores = self.tabla.item(index,"values")
56             id = valores[0]
57             usuario = valores[1]
58             if usuario == self.usuario:
59                 self.borrarcaja()
60                 messagebox.showerror('Error','No puedes eliminar tu propio usuario')
61             else:
62                 con = sqlite3.connect("usuarios.db")
63                 cursor = con.cursor()
64                 cursor.execute("DELETE FROM usuarios WHERE id=? ", (id,))
65                 cursor.execute("UPDATE usuarios SET id = id - 1 WHERE id > ? ", (id,))
66                 cursor.execute("DELETE FROM sqlite_sequence WHERE name='usuarios'")
67                 con.commit()
68                 con.close()
69                 self.Actualizartabla()
70                 self.borrarcaja("Usuario borrado correctamente")
71         except:
72             messagebox.showerror('Error','Elige un usuario')
73
74     def salir(self):
75         self.dos.destroy()
76         self.ven.destroy()
77
78     def mostrarTabla(self):
79         con = sqlite3.connect("usuarios.db")
80         cursor = con.cursor()
81         cursor.execute("SELECT * FROM usuarios")
82         for row in cursor:
83             self.tabla.insert("",END,values=row)
84         con.close()
85
86     def Mostrar(self):
87         columnas = ("ID","USUARIO","PASSWORD")
88         self.tabla = ttk.Treeview(self.dos,columns=columnas,show='headings')
89         self.tabla.place(x=10,y=100,width=350,height=190)
90         for col in columnas:
91             self.tabla.heading(col, text=col)
92             self.tabla.column(col,anchor="center",width=30)
93
94         scrollY = ttk.Scrollbar(self.dos,orient='vertical',command=self.tabla.yview)
95         scrollX = ttk.Scrollbar(self.dos,orient='horizontal',command=self.tabla.xview)
96         scrollY.place(x=360,y=98,height=20)
97         scrollX.place(x=10,y=280,width=350)
98
99         Label(self.dos,text="Escribe el usuario").place(x=10,y=10)
100        self.usu = Entry(self.dos)
101        self.usu.place(x=10,y=30)
102        self.usu.place(x=10,y=30)
103
104        Label(self.dos,text="Escribe el password").place(x=150,y=10)
105        self.pas = Entry(self.dos)
106        self.pas.place(x=150,y=30)
107        self.tabla.bind("<>TreeviewSelect>",self.Seleccionfila)
108
109        self.borrarcaja(self.mensaje)
110        self.usu.delete(0,END)
111        self.pas.delete(0,END)
112        if mensaje:
113            messagebox.showinfo('Listo',mensaje)
114
115    def Regresar(self):
116        self.dos.destroy()
117        self.ven.deiconify()
118
119    if __name__ == '__main__':
120        crearBaseDatos()
121        master = Tk()
122        app = Ventana(master)
123        app.Inicio()
124        master.mainloop()
125

```





Programa 2

Usuario

admin

Password

Validar Cerrar

Programa 1

Archivo

Escribe el usuario Escribe el password Bienvenido admin

Listo

Usuario agregado correctamente

Aceptar

ID	USUARIO	PASSWORD
1	admin	12345
2	Supervisor	12345
3	Jefe de area	12345
4	diego	112345

Programa 1

Archivo

Agregar Modificar Eliminar Salir

Escribe el password Bienvenido admin

ID	USUARIO	PASSWORD
1	admin	12345
2	Supervisor	12345
3	Jefe de area	12345
4	diego	112345





Programa 3: El programa es un sistema CRUD de productos hecho en Python con Tkinter y SQLite. Permite agregar, consultar, modificar y eliminar productos de una tienda, manejando dos tablas relacionadas (*productos* y *almacen*) mediante un INNER JOIN.

```

1  def Eliminarproducto(self):
2      try:
3          self.index = self.tabla.selection()[0]
4      except:
5          return
6      valores = self.tabla.item(self.index,"values")
7      id = valores[0]
8      con = sqlite3.connect("Tienda.db")
9      cursor = con.cursor()
10     cursor.execute("DELETE FROM productos WHERE id=?",(id))
11     cursor.execute("DELETE FROM almacen WHERE id=?",(id))
12     con.commit()
13     con.close()
14     self.Actualizartabla()
15     self.Limpiarcajass()
16     self.agregar.config(state='normal')
17     self.modificar.config(state='disabled')
18     self.eliminar.config(state='disabled')
19
20    def Seleccionfila(self,event):
21        try:
22            self.index = self.tabla.selection()[0]
23        except:
24            return
25
26        valores = self.tabla.item(self.index,"values")
27        self.producto.delete(0,END)
28        self.descri.delete(0,END)
29        self.precio.delete(0,END)
30        self.contidad.delete(0,END)
31        self.producto.insert(0,valores[2])
32        self.descri.insert(0,valores[4])
33        self.precio.insert(0,valores[3])
34        self.contidad.insert(0,valores[5])
35        self.agregar.config(state='disabled')
36        self.modificar.config(state='normal')
37        self.eliminar.config(state='normal')
38
39    def MostrarDatos(self):
40        con = sqlite3.connect("Tienda.db")
41        cursor = con.cursor()
42        cursor.execute("""
43            SELECT productos.id,
44                  productos.codigo,
45                  productos.producto,
46                  productos.precio,
47                  almacen.descripcion,
48                  almacen.stock
49                  FROM productos
50                  INNER JOIN almacen
51                  ON productos.codigo = almacen.codigodelproducto
52        """)
53        for i in cursor.fetchall():
54            self.tabla.insert("",END,values=i)
55        con.commit()
56        con.close()

```





```

 1  from tkinter import *
 2  from tkinter import messagebox
 3  from tkinter import ttk
 4  import tkinter as tkn
 5  import sqlite3
 6  import random
 7  def crearBaseDatos():
 8      con = sqlite3.connect("Tienda.db")
 9      cursor = con.cursor()
10      cursor.execute("""CREATE TABLE IF NOT EXISTS productos(
11          id INTEGER PRIMARY KEY AUTOINCREMENT,
12          codigo TEXT NOT NULL,
13          producto TEXT NOT NULL,
14          precio REAL NOT NULL)""")
15
16      cursor.execute("""CREATE TABLE IF NOT EXISTS almacen(
17          id INTEGER PRIMARY KEY AUTOINCREMENT,
18          codigodelproducto TEXT NOT NULL,
19          descripcion TEXT NOT NULL,
20          stock INTEGER NOT NULL)""")
21
22      con.commit()
23      con.close()
24
25  class Principal():
26      def __init__(self,master):
27          self.ven = master
28          self.ven.title('Programa 2')
29          ancho = 580
30          alto = 380
31          ventana_ancho = self.ven.winfo_screenwidth()
32          ventana_alto = self.ven.winfo_screenheight()
33          x = (ventana_ancho // 2) - (ancho // 2)
34          y = (ventana_alto // 2) - (alto // 2)
35          self.ven.geometry(f'{ancho}x{alto}+{x}+{y}')
36          self.index = -1
37
38      def Inicio(self):
39          self.us = Label(self.ven,text='CRUD DE PRODUCTOS').place(x=5,y=5)
40          Label(self.ven,text='Producto').place(x=10,y=25)
41          self.producto = Entry(self.ven)
42          self.producto.place(x=10,y=50)
43
44          Label(self.ven,text='Descripcion').place(x=150,y=25)
45          self.descri = Entry(self.ven)
46          self.descri.place(x=150,y=50)
47
48          Label(self.ven,text='Precio').place(x=290,y=25)
49          self.precio = Entry(self.ven)
50          self.precio.place(x=290,y=50)
51
52          Label(self.ven,text='Cantidad').place(x=430,y=25)
53          self.contidad = Entry(self.ven)
54          self.contidad.place(x=430,y=50)
55
56          # self.us.place(x=400,y=1)
57          columnas = ("ID","CODIGO","PRODUCTO","PRECIO","DESCRIPCION","STOCK")
58          self.tabla = ttk.Treeview(self.ven,columns=columnas,show='headings')
59          self.tabla.place(x=10,y=100,width=420,height=190)
60          for col in columnas:
61              self.tabla.heading(col, text=col)
62              self.tabla.column(col, anchor='center', width=30)
63          scrollx = ttk.Scrollbar(self.ven, orient='vertical', command=self.tabla.yview)
64          scrollx.place(x=430,y=98,height=210)
65          scrollx.place(x=10,y=280,width=420)
66          self.agregar = Button(self.ven, text='Agregar', width=10, state='normal', command=self.Agregarproducto)
67          self.agregar.place(y=300,x=20)
68          self.modificar = Button(self.ven, text='Modificar', width=10, state='disabled', command=self.ModificarProducto)
69          self.modificar.place(y=300,x=130)
70          self.eliminar = Button(self.ven, text='Eliminar', width=10, state='disabled', command=self.Eliminarprodcto)
71          self.eliminar.place(y=300,x=240)
72          self.tabla.bind('<<TreeviewSelect>>',self.Seleccionfila)
73          self.MostrarDatos()
74
75      def ModificarProducto(self):
76          try:
77              self.index = self.tabla.selection()[0]
78          except:
79              return
80          id = valores[0]
81          p = self.producto.get()
82          pre = self.precio.get()
83          d = self.precio.get()
84          can = self.contidad.get()
85          valores = self.tabla.item(self.index,"values")
86          if len(p) !=0 and len(pre) !=0 and len(d) !=0 and len(can) !=0:
87              codigo = p[0:2].upper() + str(random.randint(0,100)) + d[0].upper()
88              con = sqlite3.connect("Tienda.db")
89              cursor = con.cursor()
90              cursor.execute("UPDATE productos SET codigo=?, producto=?, precio=? WHERE id=?", (codigo,p,pre,id))
91              cursor.execute("UPDATE almacen SET codigodelproducto=?, descripcion=?, stock=? WHERE id=?", (codigo,d,can,id))
92              con.commit()
93              con.close()
94              self.limpiarcajas()
95              self.Actualizartabla()
96          else:
97              messagebox.showerror('Error','Faltan datos')
98

```





```

 1  def Actualizartabla(self):
 2      for i in self.tabla.get_children():
 3          self.tabla.delete(i)
 4      self.MostrarDatos()
 5
 6  def verificar(self):
 7      p = self.producto.get()
 8      d = self.descri.get()
 9      pr = self.precio.get()
10      s = self.contidad.get()
11      if len(p) !=0 and len(d) !=0 and len(pr) !=0 and len(s) !=0:
12          con = sqlite3.connect("tienda.db")
13          cursor = con.cursor()
14          cursor.execute()
15          cursor.execute("SELECT producto FROM productos WHERE producto = ? ",(p,))
16          resultado = cursor.fetchall()
17          con.commit()
18          con.close()
19          if not resultado:
20              return True
21          else:
22              return False
23
24  def Agregarproducto(self):
25      if not self.verificar():
26          messagebox.showerror('Mensaje','Ya existe')
27      p = self.producto.get()
28      d = self.descri.get()
29      pr = self.precio.get()
30      s = self.contidad.get()
31      if len(p) !=0 and len(d) !=0 and len(pr) !=0 and len(s) !=0:
32          con = sqlite3.connect("Tienda.db")
33          cursor = con.cursor()
34          cursor.execute("UPDATE productos SET precio=? WHERE id=? ", ())
35          cursor.execute("UPDATE alamacen SET stock=? WHERE id=? ", ())
36          con.commit()
37          con.close()
38      else:
39          prf = 0.0
40          st = 0
41          p = self.producto.get()
42          d = self.descri.get()
43          pr = self.precio.get()
44          s = self.contidad.get()
45          if len(p) !=0 and len(d) !=0 and len(pr) !=0 and len(s) !=0:
46              prf = float(pr)
47              st = int(s)
48              codigo = p[0:2].upper() + str(random.randint(0,100)) + d[0].upper()
49              con = sqlite3.connect("Tienda.db")
50              cursor = con.cursor()
51              cursor.execute("INSERT INTO productos (codigo,producto,precio) VALUES (?, ?, ?)", (codigo,p,prf))
52              cursor.execute("INSERT INTO alamacen (codigodelproducto,descripcion,stock) VALUES (?, ?, ?)", (codigo,d,st))
53              con.commit()
54              con.close()
55              self.limpiarcajas('Producto agregado correctamente')
56              self.Actualizartabla()
57      else:
58          messagebox.showerror('Error','Faltan datos')
59
60  def Limpiarcajas(self,mensaje):
61      self.producto.delete(0,END)
62      self.descri.delete(0,END)
63      self.precio.delete(0,END)
64      self.contidad.delete(0,END)
65      messagebox.showinfo(' ',mensaje)
66
67  def Limpiarcajass(self):
68      self.producto.delete(0,END)
69      self.descri.delete(0,END)
70      self.precio.delete(0,END)
71      self.contidad.delete(0,END)
72
73
74  if __name__ == '__main__':
75      crearBaseDatos()
76      master = Tk()
77      app = Principal(master)
78      app.Inicio()
79      master.mainloop()
80

```





Practica 2 Parcial 3

— □ ×

CRUD DE PRODUCTOS

Producto	Descripcion	Precio	Cantidad
----------	-------------	--------	----------

ID	CODIGO	PRODUCTO	PRECIO	DESCRIPCION	STOCK
1	CO82P	coca	14.0	polvo	7
2	CO52G	coca	14.0	gramo	2

[Agregar](#)

[Modificar](#)

[Eliminar](#)





Tarea 1: El código realiza una consulta a la base de datos usando SQL para obtener información de productos y su respectivo inventario. Utiliza una sentencia INNER JOIN para unir la tabla *productos* con la tabla *almacen* mediante el código de producto, y devuelve datos como el ID, código, nombre del producto, precio, descripción del almacén y stock disponible.

```

1  from tkinter import *
2  from tkinter import messagebox
3
4  class Principal():
5      def __init__(self,master):
6          self.ven = master
7          self.ven.title('Tarea 1')
8          ancho = 250
9          alto = 200
10         ventana_alto = self.ven.winfo_screenmmwidth()
11         ventana_ancho = self.ven.winfo_screenmmwidth()
12         x = (ventana_alto // 2) - (ancho // 2)
13         y = (ventana_ancho // 2) - (alto // 2)
14         self.ven.geometry(f'{ancho}x{alto}+{x+550}+{y+150}')
15
16     def Inicio(self):
17         Label(self.ven,text='Ingrese un numero').place(x=70,y=10)
18         self.n1 = Entry(self.ven)
19         self.n1.place(x=63,y=50)
20         Button(self.ven,text='Enviar a la lista',command=self.Envia).place(x=80,y=100)
21         Button(self.ven,text='Cerrar',command=self.Cerrar).place(x=100,y=140)
22
23     def Cerrar(self):
24         self.ven.destroy()
25
26     def Envia(self):
27         try:
28             n1 = int(self.n1.get())
29             self.n1.delete(0,END)
30             self.ven.withdraw()
31             otra = Toplevel(self.ven)
32             Ventanados(otra,self.ven,n1)
33         except ValueError:
34             messagebox.showerror('Error','Introduce solo numeros')
35             self.n1.delete(0,END)
36
37     class Ventanados():
38         def __init__(self,master,ven,a):
39             self.n1 = a
40             self.dos = master
41             self.dos.title('Tarea 1')
42             ancho = 250
43             alto = 200
44             ventana_alto = self.dos.winfo_screenmmwidth()
45             ventana_ancho = self.dos.winfo_screenmmwidth()
46             x = (ventana_alto // 2) - (ancho // 2)
47             y = (ventana_ancho // 2) - (alto // 2)
48             self.dos.geometry(f'{ancho}x{alto}+{x+550}+{y+150}')
49             Label(self.dos,text='Numeros').place(x=150,y=10)
50             self.lista = Listbox(self.dos, height=10, width=15, bg='grey', activestyle="dotbox", fg="Black")
51             self.lista.place(x=10, y=10)
52             Button(self.dos,text='Regresar',command=self.Regresar).place(x=155,y=40)
53             self.ven = ven
54             #-----
55             for i in range(self.n1):
56                 self.lista.insert(self.lista.size(),self.n1)
57
58         def Regresar(self):
59             self.lista.delete(0,END)
60             self.dos.destroy()
61             self.ven.deiconify()
62
63     if __name__=='__main__':
64         master = Tk()
65         app = Principal(master)
66         app.Inicio()
67         master.mainloop()

```



