

Tecnológico José Mario molina Pasquel y Henríquez UA Zapotlanejo

Manual del programador



Carrera: Ingeniería en Informática

Materia: Estructura de datos

Docente: Osvaldo Rene Rojo Roa

Alumno: Oscar Alejandro Alvarez Aceves-240111676



INDICE

PARCIAL 1	4
Programa 1: Pedir un número, calcular la raíz y elevar al cuadrado	4
Programa 2: Arreglos y listas	5
Programa 3: Ingresar datos a arreglo	6
Programa 4: datos en arreglos y listas.....	7
Tarea 1: Ingresar datos a un arreglo o una lista	8
Programa 5: resolución tarea 1	9
Programa 6: valores en listas con validaciones	10
Programa 7: información de 5 personas	11
Programa 8: Mayúsculas, minúsculas y espacios en una cadena	13
Tarea 2: lista de cadenas con condiciones.....	14
Programa 9: Resolución tarea 2	15
Repaso 1	16
Repaso 2	17
Repaso 3	18
Parcial 2	19
Programa 1 almacenamiento en lista.....	19
Programa 2 calificaciones y promedio validados	20
Programa 3 inicio de sesión con ventanas	21
Programa 4 inicio de sesión con ventanas y PACK	22
Programa 5 promedio de números con PLACE	23
Programa 6 ventanas con GRID	24
Programa 7 Uso de la biblioteca random	25
Tarea 1 ordenamiento de calificaciones messagebox	26
Tarea 1.1 ordenamiento de calificaciones consola.....	27
solución examen unidad 1.....	28
Validaciones de números y cálculo de promedio	29
Repaso	30
Validación repaso	31
Preexamen.....	32

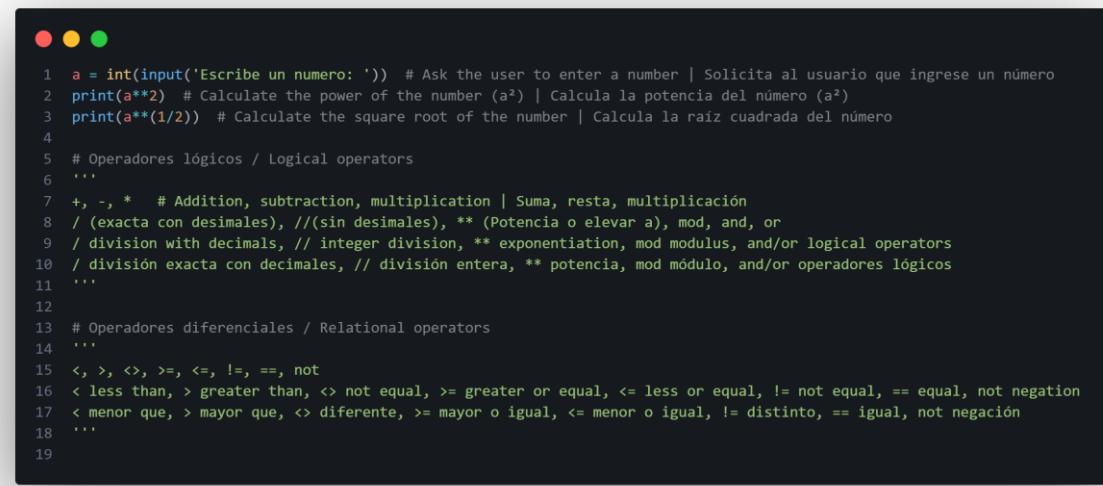


Validaciones preexamen.....	33
Parcial 3	34
Programa 1	34



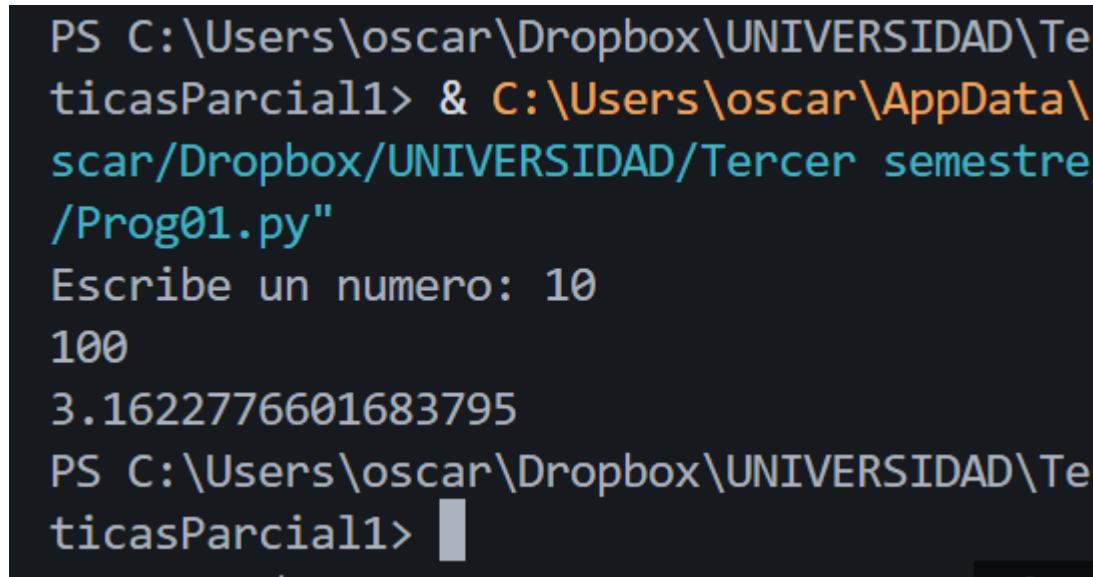
PARCIAL 1

Programa 1: Pedir un número, calcular la raíz y elevar al cuadrado



```
1 a = int(input('Escribe un numero: ')) # Ask the user to enter a number | Solicita al usuario que ingrese un número
2 print(a**2) # Calculate the power of the number (a2) | Calcula la potencia del número (a2)
3 print(a**(1/2)) # Calculate the square root of the number | Calcula la raíz cuadrada del número
4
5 # Operadores lógicos / Logical operators
6 ...
7 +, -, * # Addition, subtraction, multiplication | Suma, resta, multiplicación
8 / (exacta con decimales), //(sin decimales), ** (Potencia o elevar a), mod, and, or
9 / division with decimals, // integer division, ** exponentiation, mod modulus, and/or logical operators
10 / división exacta con decimales, // división entera, ** potencia, mod módulo, and/or operadores lógicos
11 ...
12
13 # Operadores diferenciales / Relational operators
14 ...
15 <, >, <>, >=, !=, ==, not
16 < less than, > greater than, <> not equal, >= greater or equal, <= less or equal, != not equal, == equal, not negation
17 < menor que, > mayor que, <> diferente, >= mayor o igual, <= menor o igual, != distinto, == igual, not negación
18 ...
19
```

Ilustración 1 Código Programa 1



```
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\TécnicasParcial1> & C:\Users\oscar\AppData\Local\Dropbox\UNIVERSIDAD\Tercer semestre\Prog01.py"
Escribe un numero: 10
100
3.1622776601683795
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\TécnicasParcial1>
```

Ilustración 2 ejecución programa 1



Programa 2: Arreglos y listas

Este programa muestra las diferencias entre un arreglo y una lista, también enseña la manera correcta de inicializarlas

Ilustración 3 Código programa 2

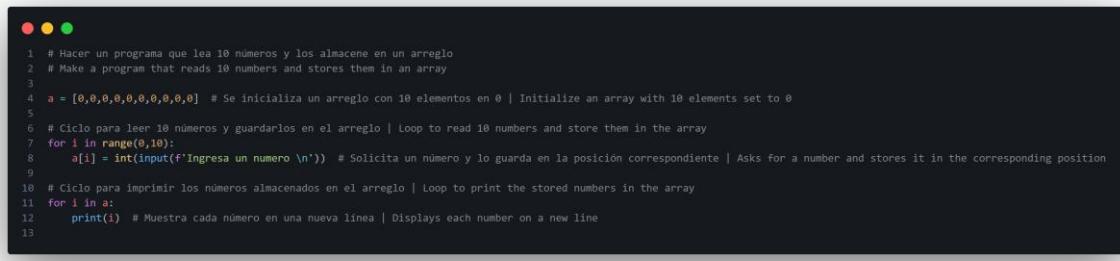
```
C:\Users\oscar\AppData\Local\Microsoft\Window  
mestre/Estructura de datos/Primer semestre/Pr  
B es mayor  
[10]  
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer
```

Ilustración 4 Ejecución programa 2



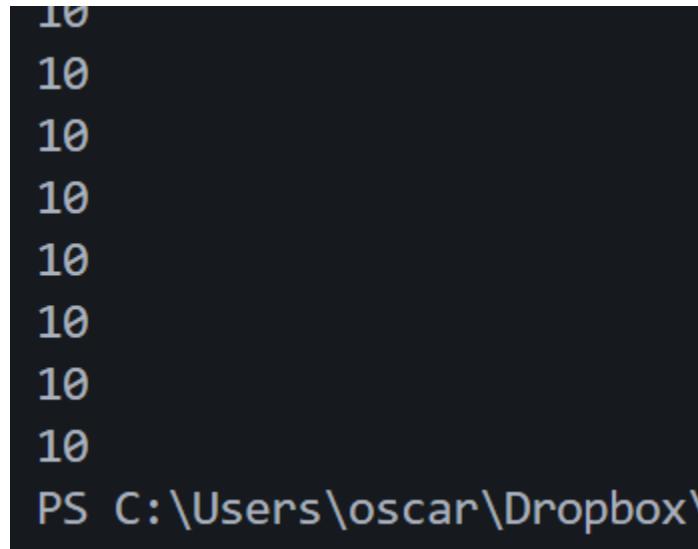
Programa 3: Ingresar datos a arreglo

Este programa enseña como añadir valores en un arreglo respetando la posición de entrada



```
1 # Hacer un programa que lea 10 números y los almacene en un arreglo
2 # Make a program that reads 10 numbers and stores them in an array
3
4 a = [0,0,0,0,0,0,0,0] # Se inicializa un arreglo con 10 elementos en 0 | Initialize an array with 10 elements set to 0
5
6 # Ciclo para leer 10 números y guardarlos en el arreglo | Loop to Read 10 numbers and store them in the array
7 for i in range(0,10):
8     a[i] = int(input('Ingresa un numero \n')) # Solicita un número y lo guarda en la posición correspondiente | Asks for a number and stores it in the corresponding position
9
10 # Ciclo para imprimir los números almacenados en el arreglo | Loop to print the stored numbers in the array
11 for i in a:
12     print(i) # Muestra cada número en una nueva linea | Displays each number on a new line
13
```

Ilustración 5 Código programa 3



```
10
10
10
10
10
10
10
10
10
PS C:\Users\oscar\Dropbox\
```

Ilustración 6 ejecución programa 3



Programa 4: datos en arreglos y listas

Este programa enseña una forma básica de validar que los datos que se ingresaran a una lista son del tipo entero, al final mostrara el total de la suma de los datos aceptados

```

1 # Hacer un programa que lea 10 números y los almacene en una lista
2 # Create a program that reads 10 numbers and stores them in a list
3
4 a = [] # Lista donde se guardarán los números | List where the numbers will be stored
5 s = 0 # Variable para almacenar la suma | Variable to store the sum
6 n = 0 # Contador para controlar la cantidad de números | Counter to control how many numbers are entered
7 numeros = "0,1,2,3,4,5,6,7,8,9"; # Define el rango de datos que se pueden usar | Defines the range of allowed digits
8
9 # Ciclo while que continúa hasta que se ingresen 10 números válidos | While loop that continues until 10 valid numbers are entered
10 while(n < 10):
11     b = input('Escribe un numero: ') # Solicita un número al usuario | Asks the user for a number
12     x = 0 # Contador de dígitos válidos | Counter for valid digits
13
14     for i in b: # Recorre cada carácter ingresado | Iterates through each entered character
15         # if (ord(i) >=48 and ord(i)<=57): # El ord se utiliza para obtener el valor ASCII | The ord function is used to get the ASCII value
16         if i in numeros: # Verifica si el carácter es un número válido | Checks if the character is a valid number
17             x += 1 # Incrementa el contador si es válido | Increments the counter if valid
18
19     if len(b) == x: # Compara si todos los caracteres son números | Checks if all characters are valid digits
20         a.append(int(b)) # Convierte a entero el valor y lo agrega a la lista | Converts the value to integer and appends it to the list
21         n += 1 # Incrementa el contador | Increments the counter
22     else:
23         print('El valor no es numero') # Muestra un mensaje si hay caracteres inválidos | Displays a message if invalid characters are detected
24
25 # Imprime todos los números de la lista | Prints all the numbers stored in the list
26 for i in a:
27     print(i) # Muestra cada número | Displays each number
28     s += 1 # Suma cada número a la variable acumuladora | Adds each number to the accumulator
29
30 # Muestra la suma total de los números ingresados | Displays the total sum of the entered numbers
31 print(f'La suma es: {s}')

```

Ilustración 7 Código programa 4

```

10
10
10
10
10
10
10
10
La suma es: 100
PS C:\Users\oscar\Dropbox\UNIVE

```

Ilustración 8 ejecución programa 4



Tarea 1: Ingresar datos a un arreglo o una lista

Este programa enseña una manera básica de validar si un dato es numérico o de caracteres, si es numérico lo almacenara en un arreglo, por otro lado, si es carácter lo guardara en una lista, el programa hará esto hasta que las longitudes del arreglo y lista sumen un total de 10

```

1 #Hacer un programa que lea 10 datos, si el dato es un numero se almacenara en un arreglo
2 # si es un caracter o caracteres se metera en una lista, cuando finalice el programa
3 # nos mostrara cuantos elementos numericos y cuantos caracteres hay en cada estructuras
4
5 car = 0 # EN: Variable declared but unused / ES: Variable declarada pero no utilizada
6 Arreglo = 0 # EN: Counter for total entered elements / ES: Contador para los elementos totales ingresados
7 n = 0 # EN: Counter for total entered elements / ES: Contador para los elementos ingresados
8 a = [0,0,0,0,0,0,0,0,0] # EN: Initialize list 'a' with one numeric element (10) / ES: Inicializa la lista 'a' con un elemento numérico (10)
9 l = [] # EN: Initialize empty list 'l' to store alphabetic characters / ES: Inicializa la lista vacía 'l' para guardar caracteres alfabéticos
10
11 while (n < 10): # EN: Loop until 10 valid elements are entered / ES: Bucle hasta que se ingresen 10 elementos válidos
12     b = input('Ingresa un dato: ') # EN: Request input from user / ES: Solicita un dato al usuario
13
14     if b.isdigit(): # EN: Check if input is numeric / ES: Verifica si el dato ingresado es numérico
15         a[Arreglo]=b # EN: Add numeric input to list 'a' / ES: Agrega el dato numérico a la lista 'a'
16         Arreglo +=1 # EN: Increase numeric counter / ES: Incrementa el contador de numéricos
17         n += 1 # EN: Increment counter / ES: Incrementa el contador
18     elif b.isalpha(): # EN: Check if input is alphabetic / ES: Verifica si el dato ingresado es alfabetico
19         l.append(b) # EN: Add alphabetic input to list 'l' / ES: Agrega el dato alfabetico a la lista 'l'
20         n += 1 # EN: Increment counter / ES: Incrementa el contador
21     else:
22         print('No valido') # EN: Show error message for invalid input / ES: Muestra mensaje de error para datos no válidos
23
24 # ---- Final results ----
25 print(f'El total de elementos numericos guardados es {(Arreglo)}')
26 # EN: Show total numeric elements stored / ES: Muestra el total de elementos numéricos guardados
27 print(f'El total de caracteres guardados es {len(l)})')
28 # EN: Show total alphabetic elements stored / ES: Muestra el total de caracteres guardados
29 """print("Elementos numericos guardados (a)")
30 # EN: Display list of numeric elements / ES: Muestra la lista de elementos numéricos
31 print(f'Elementos de caracteres guardados {l}')
32 # EN: Display list of alphabetic elements / ES: Muestra la lista de elementos alfabéticos"""
33

```

Ilustración 9 Código tarea 1

```

Ingresa un dato: hola
Ingresa un dato: que
Ingresa un dato: tal
Ingresa un dato: 3
Ingresa un dato: 2
El total de elementos numericos guardados es 6
El total de caracteres guardados es 4
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer semest
ricasParcial1>

```

Ilustración 10 ejecución tarea 1



Programa 5: resolución tarea 1

Este programa enseña una manera alternativo de resolver la problemática de la tarea 1

```

1 arr = [0,0,0,0,0,0,0,0,0] # Arreglo para almacenar números | Array to store numbers
2 car = [] # Lista para almacenar cadenas | List to store strings
3 c = 0 # Contador para la posición del arreglo | Counter for array position
4 c2 = 0 # Contador de números válidos en el arreglo | Counter for valid numbers in the array
5
6 # Bucle para leer hasta 10 datos | Loop to read up to 10 inputs
7 while (True):
8     a = input('Escribe un dato o valor \n') # Solicita un dato al usuario | Asks the user for a value
9
10    if a.isdigit(): # Verifica si es un número | Checks if the input is a number
11        arr[c] = int(a) # Convierte a entero y lo guarda en el arreglo | Converts to integer and stores in the array
12    elif a.isalpha(): # Verifica si es solo letras | Checks if the input contains only letters
13        car.append(a) # Agrega la cadena a la lista | Appends the string to the list
14
15    c += 1 # Incrementa el contador | Increments the counter
16    if c >= 10: # Sale del bucle después de 10 entradas | Exits the loop after 10 inputs
17        break
18
19 # Cuenta cuántos números válidos hay en el arreglo | Counts how many valid numbers are in the array
20 for i in arr:
21     if i != 0:
22         c2 += 1
23
24 print(f'El arreglo tiene {c2}') # Muestra la cantidad de números | Displays the number of numbers
25 print(f'La lista tiene {len(car)}') # Muestra la cantidad de cadenas | Displays the number of strings
26
27 print(car) # Imprime la lista de cadenas | Prints the list of strings
28 print(arr) # Imprime el arreglo de números | Prints the array of numbers
29
30

```

Ilustración 11 Código programa 5

```

Escribe un dato o valor
mundo
Escribe un dato o valor
que
El arreglo tiene 5
La lista tiene 5
['oscar', 'ismael', 'hola', 'mundo', 'que']
[0, 22, 0, 123, 124, 22, 1, 0, 0, 0]
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer se
φ OscarAlvarez262 (6 days ago)

```

Ilustración 12 ejecución programa 5



Programa 6: valores en listas con validaciones

Este programa enseña el uso de funciones para realizar una validación si un dato ingresado es tipo numérico o carácter almacenando en un arreglo o lista respectivamente, mostrando al final el total y cuales datos fueron almacenados

```

1 def hola(): # Definición de método o función | Method or function definition
2     c = 0 # Contador para controlar la posición en el arreglo | Counter to control the position in the array
3     while (True): # Bucle infinito hasta que se ingrese 10 datos | Infinite loop until 10 inputs are given
4         a = input('Escribe un dato o valor \n') # Solicitud un dato al usuario | Asks the user for a value
5
6         if a.isdigit(): # Verifica si la entrada es un número | Checks if the input is a number
7             arr[c] = int(a) # Convierte a entero y lo guarda en el arreglo | Converts to integer and stores it in the array
8         elif a.isalpha(): # Verifica si la entrada es solo letras | Checks if the input is alphabetic
9             car.append(a) # Agrega la cadena a la lista car | Appends the string to the car list
10
11         c += 1 # Incrementa el contador | Increments the counter
12
13     if c >= 10: # Condición para detener el ciclo después de 10 entradas | Stops the loop after 10 inputs
14         break
15
16     resultados() # Llama a la función para mostrar los resultados | Calls the function to display results
17
18
19 def resultados(): # Definición de método o función | Method or function definition
20     c2 = 0 # Contador de valores numéricos válidos en el arreglo | Counter for valid numeric values in the array
21     print(f'La lista tiene {len(car)}') # Muestra la cantidad de cadenas guardadas | Displays the number of stored strings
22
23     for i in arr: # Recorre el arreglo numérico | Iterates over the numeric array
24         if i != 0: # Verifica si el valor no es cero | Checks if the value is not zero
25             c2 += 1 # Incrementa el contador | Increments the counter
26
27     print(f'El arreglo tiene {c2}') # Muestra la cantidad de números guardados | Displays the number of stored numbers
28     print(car) # Imprime la lista de cadenas | Prints the list of strings
29     print(arr) # Imprime el arreglo de números | Prints the array of numbers
30
31
32 arr = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] # Arreglo inicializado con 10 ceros | Array initialized with 10 zeros
33 car = [] # Lista para almacenar cadenas | List to store strings
34
35 if __name__ == "__main__": # Método principal | Main method
36     hola() # Llama a la función principal | Calls the main function
37

```

Ilustración 13 código programa 6



```

tal
Escribe un dato o valor
133
La lista tiene 6
El arreglo tiene 4
['hola', 'hola', 'hola', 'hola', 'que', 'tal']
[0, 0, 3, 0, 10, 0, 45, 0, 0, 133]
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer sem
ticasParcial1> █

```

Ilustración 14 ejecución programa 6

Programa 7: información de 5 personas

Este programa enseña el cómo almacenar múltiples variables en una posición de una lista

```

● ● ●
1 # Hacer un programa que lea nombre, edad y sexo de 5 personas
2 # Make a program that reads name, age, and gender of 5 people
3
4 def lista(): # Definición de función | Function definition
5     c = 0 # Contador para controlar cuántas personas se registran | Counter to control how many people are registered
6     while True: # Bucle infinito hasta que se ingresen 5 personas | Infinite loop until 5 people are entered
7         n = input('Ingresa un nombre \n') # Solicita el nombre | Asks for the name
8         e = input('Ingresa su edad \n') # Solicita la edad | Asks for the age
9         g = input('Ingresa su sexo H o M \n').upper() # Solicita el sexo y lo convierte a mayúsculas | Asks for gender and converts it to uppercase
10
11     # Se guarda la información en la lista lis1 como una cadena | Stores the information in lis1 as a string
12     lis1.append(n + ', ' + e + ', ' + g)
13
14     c += 1 # Incrementa el contador | Increments the counter
15
16     if c >= 5: # Si ya se ingresaron 5 personas, sale del bucle | Stops the loop after 5 people are entered
17         break
18
19     resultado() # Llama a la función para mostrar resultados | Calls the function to display results
20
21
22 def resultado(): # Definición de función para mostrar resultados | Function definition to display results
23     print(lis1) # Imprime la lista con los datos registrados | Prints the list with the stored data
24
25
26 lis1 = [] # Lista para almacenar nombre, edad y sexo | List to store name, age, and gender
27 lis2 = [] # Lista no utilizada actualmente | List not used currently
28
29 # Punto de inicio del programa | Program entry point
30 if __name__ == "__main__":
31     lista() # Llama a la función principal | Calls the main function
32

```

Ilustración 15 código programa 7



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Ingresá un nombre
sandra
Ingresá su edad
20
Ingresá su sexo H o M
M
['jose, 33, H', 'Maria, 12, M', 'oscar, 23, H', 'Diego, 19, H', 'sandra, 20, M']
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer semestre\Estructura de datos\Primer semestre\Práctica 1\src>
```

Ilustración 16 ejecución programa 7



Programa 8: Mayúsculas, minúsculas y espacios en una cadena

Este programa enseña la manera de hacer validaciones de características especiales incluyendo especificaciones y reglas en los datos que podrán ser almacenados en una lista

```

1 # Hacer un programa que lea una cadena y que muestre en pantalla
2 # cuántos números tiene, cuántas mayúsculas, cuántas minúsculas
3 # y cuántos espacios
4 # Make a program that reads a string and displays how many numbers, uppercase letters,
5 # lowercase letters, and spaces it contains
6
7 def inicio(): # Definición de función | Function definition
8     numeros = "0123456789" # Cadena de números válidos | String containing valid numbers
9     cn = 0 # Contador de números | Counter for numbers
10    e = 0 # Contador de espacios | Counter for spaces
11    min = 0 # Contador de minúsculas | Counter for lowercase letters
12    may = 0 # Contador de mayúsculas | Counter for uppercase letters
13
14    cadena = input('Escribe una cadena \n') # Sigue la cadena al usuario | Asks the user to enter a string
15
16    # Recorre cada carácter en la cadena | Iterates over each character in the string
17    for i in cadena:
18        if i in numeros: # Verifica si el carácter es un número | Checks if the character is a number
19            cn += 1 # Incrementa el contador de números | Increments the number counter
20
21        if i == ' ': # Verifica si el carácter es un espacio | Checks if the character is a space
22            e += 1 # Incrementa el contador de espacios | Increments the space counter
23
24        if ord(i) >= 97 and ord(i) <= 122: # Verifica si es una letra minúscula | Checks if it's a lowercase letter
25            min += 1 # Incrementa el contador de minúsculas | Increments lowercase counter
26
27        if ord(i) >= 65 and ord(i) <= 90: # Verifica si es una letra mayúscula | Checks if it's an uppercase letter
28            may += 1 # Incrementa el contador de mayúsculas | Increments uppercase counter
29
30    # Muestra los resultados finales | Displays the final results
31    print(f'Los numeros son {cn},\n los espacios: {e},\n las minusculas son: {min},\n las mayusculas son: {may}')
32
33
34 # Punto de inicio del programa | Program entry point
35 if __name__ == '__main__':
36     inicio() # Llama a la función principal | Calls the main function
37
38

```

Ilustración 17 Código programa 8

```

/Prog08.py"
Escribe una cadena
Hola que tal
Los numeros son 0,
los espacios: 2,
las minusculas son: 9,
las mayusculas son: 1
PS C:\Users\oscar\Dropbox\UNIVERS
ticasParcial1> █

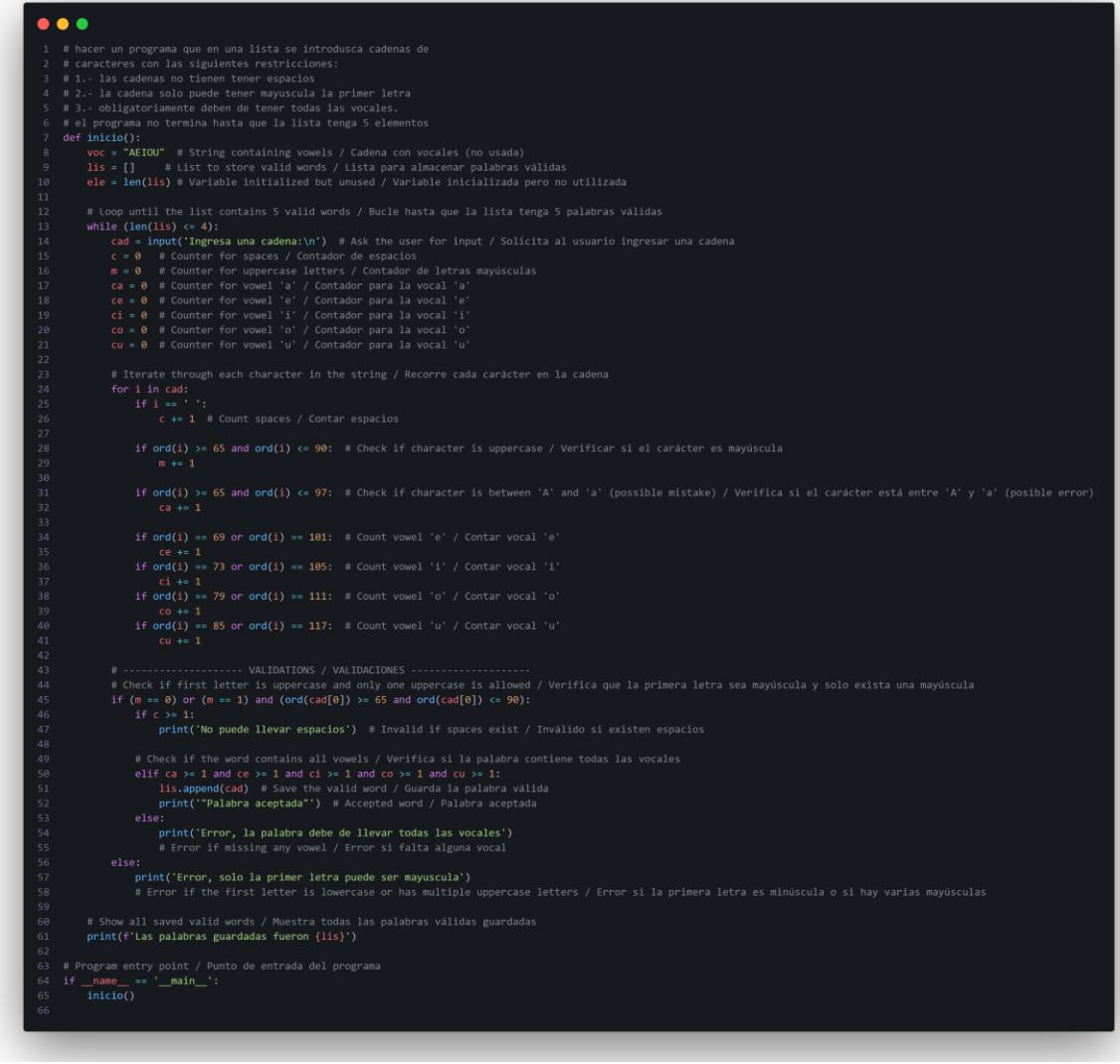
```

Ilustración 18 ejecución programa 8



Tarea 2: lista de cadenas con condiciones

Este programa enseña la manera de hacer validaciones de características especiales incluyendo especificaciones y reglas en los datos que podrán ser almacenados en una lista



```

1 # hacer un programa que en una lista se introduzca cadenas de
2 # caracteres con las siguientes restricciones:
3 # 1.- las cadenas no tienen tener espacios
4 # 2.- la cadena solo puede tener mayúscula la primer letra
5 # 3.- obligatoriamente deben de tener todas las vocales.
6 # el programa no termina hasta que la lista tenga 5 elementos
7 def inicio():
8     voc = "AEIOU" # String containing vowels / Cadena con vocales (no usada)
9     lis = [] # List to store valid words / Lista para almacenar palabras válidas
10    ele = len(lis) # Variable initialized but unused / Variable inicializada pero no utilizada
11
12    # Loop until the list contains 5 valid words / Bucle hasta que la lista tenga 5 palabras válidas
13    while (len(lis) <= 4):
14        cad = input('Ingresa una cadena:\n') # Ask the user for input / Solicita al usuario ingresar una cadena
15        c = 0 # Counter for spaces / Contador de espacios
16        m = 0 # Counter for uppercase letters / Contador de letras mayúsculas
17        ca = 0 # Counter for vowel 'a' / Contador para la vocal 'a'
18        ce = 0 # Counter for vowel 'e' / Contador para la vocal 'e'
19        ci = 0 # Counter for vowel 'i' / Contador para la vocal 'i'
20        co = 0 # Counter for vowel 'o' / Contador para la vocal 'o'
21        cu = 0 # Counter for vowel 'u' / Contador para la vocal 'u'
22
23        # Iterate through each character in the string / Recorre cada carácter en la cadena
24        for i in cad:
25            if i == ' ':
26                c += 1 # Count spaces / Contar espacios
27
28            if ord(i) >= 65 and ord(i) <= 90: # Check if character is uppercase / Verificar si el carácter es mayúscula
29                m += 1
30
31            if ord(i) >= 65 and ord(i) <= 97: # Check if character is between 'A' and 'a' (possible mistake) / Verifica si el carácter está entre 'A' y 'a' (possible error)
32                ca += 1
33
34            if ord(i) == 69 or ord(i) == 101: # Count vowel 'e' / Contar vocal 'e'
35                ce += 1
36            if ord(i) == 73 or ord(i) == 105: # Count vowel 'i' / Contar vocal 'i'
37                ci += 1
38            if ord(i) == 79 or ord(i) == 111: # Count vowel 'o' / Contar vocal 'o'
39                co += 1
40            if ord(i) == 85 or ord(i) == 117: # Count vowel 'u' / Contar vocal 'u'
41                cu += 1
42
43        # ----- VALIDACIONES / VALIDACIONES -----
44        # Check if first letter is uppercase and only one uppercase is allowed / Verifica que la primera letra sea mayúscula y solo exista una mayúscula
45        if (m == 0) or (m == 1) and (ord(cad[0]) >= 65 and ord(cad[0]) <= 90):
46            if c >= 1:
47                print('No puede llevar espacios') # Invalid if spaces exist / Inválido si existen espacios
48
49            # Check if the word contains all vowels / Verifica si la palabra contiene todas las vocales
50            elif ca >= 1 and ce >= 1 and ci >= 1 and co >= 1 and cu >= 1:
51                lis.append(cad) # Save the valid word / Guarda la palabra válida
52                print("Palabra aceptada") # Accepted word / Palabra aceptada
53            else:
54                print('Error, la palabra debe de llevar todas las vocales') # Error if missing any vowel / Error si falta alguna vocal
55            else:
56                print('Error, solo la primera letra puede ser mayúscula') # Error if the first letter is lowercase or has multiple uppercase letters / Error si la primera letra es minúscula o si hay varias mayúsculas
57
58        # Show all saved valid words / Muestra todas las palabras válidas guardadas
59        print(f'Las palabras guardadas fueron {lis}')
60
61    # Program entry point / Punto de entrada del programa
62    if __name__ == '__main__':
63        inicio()
64
65
66

```

Ilustración 19 Código tarea 2



PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
----------	--------	---------------	----------	-------

```

Eaieou
"Palabra aceptada"
Ingresa una cadena:
mamemimomu
"Palabra aceptada"
Las palabras guardadas fueron ['Murcielago', 'Aeiou', 'aaaaeeeeiiioooouu', 'Eaieou', 'mamemimo
mu']
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer semestre\Estructura de datos\Primer semestre\Prac
ticasParcial1>

```

Ilustración 20 ejecución tarea 2



Programa 9: Resolución tarea 2

Este programa enseña la manera de hacer validaciones de características especiales incluyendo especificaciones y reglas en los datos que podrán ser almacenados en una lista

```

1 def minusculas(c1): # Verifica si todos los caracteres excepto el primero son minúsculas | Checks if all characters except the first are lowercase
2     cm = 0 # Contador de minúsculas | Counter for lowercase letters
3     for i in c1[1:]: # Recorre la cadena desde el segundo carácter | Iterates over the string from the second character
4         if ord(i) >= 97 and ord(i) <= 122: # Comprueba si el carácter es minúscula | Checks if the character is lowercase
5             cm += 1
6     if cm == len(c1) - 1: # Si todas excepto la primera son minúsculas | If all except the first are lowercase
7         print("La cadenas son minusculas excepto la primera") # Mensaje de confirmación | Confirmation message
8         vocales(c1) # Llama a la función vocales para verificar que contenga todas las vocales | Calls vocales function to check all vowels
9     else:
10        print("La cadena no cumple") # Mensaje si no cumple la condición | Message if condition is not met
11
12 def vocales(cad): # Verifica si la cadena contiene todas las vocales | Checks if the string contains all vowels
13     ba = False # Bandera para la vocal 'a' | Flag for vowel 'a'
14     be = False # Bandera para la vocal 'e' | Flag for vowel 'e'
15     bi = False # Bandera para la vocal 'i' | Flag for vowel 'i'
16     bo = False # Bandera para la vocal 'o' | Flag for vowel 'o'
17     bu = False # Bandera para la vocal 'u' | Flag for vowel 'u'
18
19     if 'a' in cad or 'A' in cad:
20         ba = True
21     if 'e' in cad or 'E' in cad:
22         be = True
23     if 'i' in cad or 'I' in cad:
24         bi = True
25     if 'o' in cad or 'O' in cad:
26         bo = True
27     if 'u' in cad or 'U' in cad:
28         bu = True
29
30     # Si contiene todas las vocales, se agrega a la lista | If it contains all vowels, append to list
31     if ba and be and bi and bo and bu:
32         list.append(cad)
33
34 def leer(): # Función para leer y procesar la cadena | Function to read and process the string
35     ce = 0 # Contador de caracteres sin espacio | Counter for characters without spaces
36     nc = "" # Nueva cadena filtrada sin números | New filtered string without numbers
37     c = input('Introduce una cadena\n') # Solicitud al usuario la cadena | Asks the user for a string
38
39     # Contar caracteres que no sean espacios | Count characters that are not spaces
40     for i in c:
41         if ord(i) != 32:
42             ce += 1
43
44     if ce == len(c): # Si no hay espacios | If there are no spaces
45         if c.isalpha(): # Si la cadena es solo letras | If string contains only letters
46             minusculas(c) # Llama a la función minusculas | Calls minusculas function
47         else: # Si contiene números u otros caracteres | If it contains numbers or other characters
48             for i in c:
49                 if ord(i) >= 48 and ord(i) <= 57: # Ignora números | Ignores numbers
50                     pass
51                 else:
52                     nc += i # Construye cadena filtrada | Builds filtered string
53             print(nc) # Muestra la cadena filtrada | Displays filtered string
54             minusculas(nc) # Llama a minusculas con la cadena filtrada | Calls minusculas with filtered string
55     else:
56         print("Error, hay espacios en la cadena") # Mensaje de error si hay espacios | Error message if spaces are present
57
58     list = [] # Lista para almacenar cadenas válidas | List to store valid strings
59
60     if __name__ == '__main__': # Punto de entrada del programa | Program entry point
61         while True:
62             leer() # Llama a la función leer repetidamente | Calls leer function repeatedly
63             if len(list) >= 5: # Sale del bucle cuando la lista tiene 5 elementos válidos | Exit loop when list has 5 valid strings
64                 break
65

```

Ilustración 21 Código programa 9



Repaso 1

Este programa recopila los tipos de datos, el uso de rangos y las conversiones de datos ingresados

```

1  """instrucciones de entrada y salida"""
2  #Print() o print(f)
3  # print('Hola mundo')
4  # print(f'Hello mundo {10}')
5
6  """Entrada de datos"""
7  #input
8  # input('Escribe un numero')#Se introducen SOLO letras
9  # #Casting para convertir a valores específicos
10 # f=0.0
11 # f = float(input('Ingresa un numero con decimales'))#Solo numeros flotantes
12 # e=0
13 # e= int(input('Escribe un numero'))#SOLO numeros enteros
14 # c =120
15 # print(str(c))#Convertir a cadena
16 # vs=""
17 # v=str(c)
18 """NOTA: SOLO las variables que no se introducen por teclado se OBLIGAN a inicializarlas"""
19
20
21 # # Hacer un programa que sea nombre y precio de un producto, el programa
22 # # calculara el costo y precio de venta, el costo involucra el 12%
23 # y el iva 16%
24
25
26
27 for i in range(0, 5): # Repite el ciclo 5 veces | Loops 5 times
28     p = "" # Variable para almacenar el nombre del producto | Variable to store the product name
29     c = 0.0 # Precio bruto | Base price
30     u = 0.0 # Precio con IVA del 12% | Price with 12% tax
31     v = 0.0 # Precio de venta final con IVA + 16% | Final selling price with additional 16%
32
33     # Solicita el nombre del producto | Asks for the product name
34     p = input('Ingresa el nombre de un producto\n')
35
36     # Solicita el precio bruto del producto | Asks for the base price of the product
37     c = float(input('Ingresa el precio bruto del producto: {}{}\n'))
38
39     # Calcula el precio con el 12% agregado | Calculates the price with an additional 12%
40     u = float(c * 1.12)
41
42     # Calcula el precio final de venta con otro 16% agregado | Calculates the final price with an extra 16%
43     v = (u * 1.16)
44
45     # Muestra el precio con el 12% de incremento | Displays the price after 12% increase
46     print(f'tu costo del producto: {p}, es de ${u:.2f}') # :.2f muestra 2 decimales | :.2f shows 2 decimals
47
48     # Muestra el precio final de venta | Displays the final selling price
49     print(f'tu precio de venta del producto: {p}, es de ${v:.2f}') # :.2f muestra 2 decimales | :.2f shows 2 decimals

```

Ilustración 22 Código repaso 1

```

mestre/Estructura de datos/Primer semestre/PracticasParcial1
Ingresa el nombre de un producto
papa
Ingresa el precio bruto del producto: papa
12
tu costo del producto: papa, es de $13.44
tu precio de venta del producto: papa, es de $15.59
Ingresa el nombre de un producto

```

Ilustración 23 ejecución repaso 1



Repasso 2

Este programa enseña la manera de manipular datos tipos números ingresados para poder hacer potencias o raíces para poder realizar operaciones

```

● ● ●

1 # Hacer un programa que realice la operación de la fórmula general
2 # Make a program that performs the quadratic formula operation
3
4 a = 0 # Coeficiente A de la ecuación cuadrática | Coefficient A of the quadratic equation
5 b = 0 # Coeficiente B de la ecuación cuadrática | Coefficient B of the quadratic equation
6 c = 0 # Coeficiente C de la ecuación cuadrática | Coefficient C of the quadratic equation
7
8 p = 0 # Variable temporal para b² | Temporary variable for b squared
9 m = 0 # Variable temporal para 4ac | Temporary variable for 4ac
10 r = 0 # Discriminante b² - 4ac | Discriminant b² - 4ac
11
12 ra = 0.0 # Raíz cuadrada del discriminante | Square root of the discriminant
13 d = 0.0 # Denominador 2a | Denominator 2a
14 x1 = 0.0 # Primera solución de la ecuación cuadrática | First solution of the quadratic equation
15 x2 = 0.0 # Segunda solución de la ecuación cuadrática | Second solution of the quadratic equation
16
17
18 # Solicita los valores de A, B y C | Asks for the values of A, B, and C
19 a = int(input('Ingresa el valor de A: '))
20 b = int(input('Ingresa el valor de B: '))
21 c = int(input('Ingresa el valor de C: '))
22
23 # Calcula b² | Calculates b squared
24 p = b ** 2
25
26 # Calcula 4ac | Calculates 4ac
27 m = 4 * a * c
28
29 # Calcula el discriminante: b² - 4ac | Calculates the discriminant: b² - 4ac
30 r = p - m
31
32 # Verifica si el discriminante es positivo | Checks if the discriminant is positive
33 if r > 0:
34     print('Si se puede') # La ecuación tiene dos soluciones reales | The equation has two real solutions
35
36     # Calcula la raíz cuadrada del discriminante | Calculates the square root of the discriminant
37     ra = r ** (1/2)
38
39     # Calcula 2a | Calculates 2a
40     d = 2 * a
41
42     # Calcula las dos soluciones de la ecuación | Calculates the two solutions of the equation
43     x1 = (-b + ra) / d
44     x2 = (-b - ra) / d
45
46     # Muestra los resultados con dos decimales | Displays the results with two decimals
47     print(f'El valor de x1 es: {x1:.2f} y de x2 es: {x2:.2f}')
48
49 else:
50     # Si el discriminante es negativo, no hay soluciones reales | If the discriminant is negative, no real solutions
51     print('No se puede')
52
53
54
55
56
57

```

Ilustración 24 Código repaso 2

```

PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer semestre\Estructura de datos\Primer semestre\Práctica 1>
C:\Users\oscar\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\HJLWV9K>
Ingresá el valor de A: 10
Ingresá el valor de B: 20
Ingresá el valor de C: 4
Si se puede
El valor de x1 es: -0.23 y de x2 es: -1.77
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer semestre\Estructura de datos\Primer semestre\Práctica 1>

```

Ilustración 25 ejecución repaso 2



Repasso 3

Este programa recopila los métodos que se pueden utilizar para poder realizar validaciones, también la manera de guardar información en listas

```

1
2 # def leer():
3 #     #ORD que obtiene el ASCII del carácter
4 #     #ISALPHA para caracteres
5 #     #ISODIGIT para números
6 #     #TRY EXCEPT VALUEERROR:
7 #         a = input('Escribe un dato o valor')
8 #         validar(a)
9
10 # def validar(a):
11 #     c=0
12 #     d=0.0
13 #     try:
14 #         c= int(a)
15 #         print('Es un valor numérico sin decimales')
16 #     except ValueError:
17 #         print('No es un valor numérico sin decimales')
18 #     try:
19 #         d = float(a)
20 #         print('Es un valor numérico con decimales')
21 #     except ValueError:
22 #         print('No es un valor numérico con decimales')
23
24 # if __name__=="__main__":
25 #     leer()
26
27 # Hacer un programa que lea un dato, cualquiera que sea, y lo lo almacene en una lista, respetando su tipo de dato
28 # Make a program that reads any input and stores it in a list, preserving its data type
29
30 list = [] # Lista para almacenar los datos ingresados | List to store the entered data
31
32 def dato(): # Función para leer un dato y validarlo | Function to read a data item and validate it
33     d = input('Ingresa un dato\n') # Solicitud un dato al usuario | Asks the user for a data item
34     dato = validar(d) # Llama a la función validar para determinar el tipo de dato | Calls validar function to determine data type
35     list.append(dato) # Agrega el dato a la lista | Appends the data to the list
36
37 def validar(d): # Función para verificar el tipo de dato ingresado | Function to check the data type of the input
38     i = 0 # Variable para entero | Variable for integer
39     f = 0.0 # Variable para flotante | Variable for float
40     try:
41         i = int(d) # Intenta convertir a entero | Tries to convert to integer
42         print("Tu valor es un entero") # Mensaje si es entero | Message if input is integer
43         return i # Devuelve el entero | Returns the integer
44     except ValueError:
45         print("Tu valor no es un entero") # Mensaje si no es entero | Message if input is not integer
46     try:
47         f = float(d) # Intenta convertir a flotante | Tries to convert to float
48         print("Tu valor es un flotante") # Mensaje si es flotante | Message if input is float
49         return f # Devuelve el flotante | Returns the float
50     except ValueError:
51         print("Tu valor no es un flotante") # Mensaje si no es flotante | Message if input is not float
52
53     return d # Si no es ni entero ni flotante, devuelve el dato original | If not int or float, returns the original input
54
55 if __name__ == '__main__': # Punto de entrada del programa | Program entry point
56     while True:
57         dato() # Llama a la función dato | Calls the dato function
58         resp = input('Deseas otro S/N\n') # Pregunta si desea ingresar otro dato | Asks if user wants to enter another data item
59         if resp == 'N' or resp == 'n': # Si la respuesta es no, termina el ciclo | If the answer is no, exits the loop
60             print(list) # Muestra la lista con los datos ingresados | Displays the list with entered data
61             break
62

```

Ilustración 26 Código repaso 3

```

PS C:\Users\oscar\Dropbox\UNI
C:\Users\oscar\AppData\Local\mestre\Estructura de datos\Pr
Ingresá un dato
40
Tu valor es un entero
Deseas otro S/N

```

Ilustración 27 ejecución repaso 3



Parcial 2

Programa 1 almacenamiento en lista

Este programa permite agregar 5 calificaciones a una lista, utilizando funciones y variables globales

```

1  def inicio(num):                      # Define una función llamada inicio que recibe un parámetro num
2      # Defines a function named 'inicio' that receives one parameter 'num'
3
4      a = int(input('Escribe una calificación\n'))  # Pide al usuario que ingrese una calificación y la convierte a entero
5      # Asks the user to enter a grade and converts it to an integer
6
7      num += 1                         # Incrementa el contador de número de calificaciones
8      # Increases the counter for the number of grades entered
9
10     lista.append(a)                  # Agrega la calificación ingresada a la lista
11     # Adds the entered grade to the list
12
13     if num >= 5:                   # Si ya se ingresaron 5 calificaciones o más
14         # If 5 or more grades have been entered
15         print(lista)                # Muestra la lista completa de calificaciones
16         # Prints the full list of grades
17     else:                          # Si todavía no se han ingresado 5 calificaciones
18         # If fewer than 5 grades have been entered
19         inicio(num)                # Llama de nuevo a la función (recursión)
20         # Calls the function again (recursion)
21
22
23     lista = []                     # Crea una lista vacía para guardar las calificaciones
24     # Creates an empty list to store the grades
25
26     global num                     # Declara la variable num como global
27     # Declares the variable 'num' as global
28
29     num = 0                        # Inicializa el contador en 0
30     # Initializes the counter to 0
31
32     if __name__ == "__main__":       # Comprueba si el script se ejecuta directamente
33         # Checks if the script is being run directly
34         inicio(num)                # Llama a la función inicio() para comenzar el proceso
35         # Calls the inicio() function to start the process
36

```

Ilustración 28 Código Programa 1

```

PS C:\Users\oscar\Dropbox\UNIVERSIDAD
r semestre\Estructura de datos\Segund
Escribe una calificación
10
Escribe una calificación
12
Escribe una calificación
4
Escribe una calificación
4
Escribe una calificación
4
[10, 12, 4, 4, 4]
PS C:\Users\oscar\Dropbox\UNIVERSIDAD

```

Ilustración 29 Ejecución programa 1



Programa 2 calificaciones y promedio validados

Este programa agrega datos a una lista validando previamente desde otra clase especificando el tipo de dato a entero y obteniendo un promedio al final del ciclo de 5 datos.

```

1  From validaciones import Validacion          # Importa la clase Validacion desde el archivo validaciones.py
2  # Imports the 'Validacion' class from the 'validaciones.py' file
3
4  Val = Validacion()                          # Crea un objeto de la clase Validacion para usar sus métodos
5  # Creates an instance of the Validacion class to use its methods
6
7
8  class Principal():                         # Define una clase llamada Principal
9      # Defines a class called Principal
10
11     def __init__(self):                      # Método constructor: se ejecuta al crear un objeto de la clase
12         # Constructor method: runs when an object of the class is created
13         self.lista = []                      # Lista para almacenar las calificaciones ingresadas
14         # List to store the entered grades
15         self.num = 0                         # Contador para llevar el número de calificaciones
16         # Counter to track how many grades have been entered
17         self.a = ""                          # Variable temporal para guardar la entrada del usuario
18         # Temporary variable to store the user input
19
20     def inicio(self):                      # Método principal del programa
21         # Main method of the program
22         self.a = input('Escribe una calificación\n')
23         # Solicita al usuario que ingrese una calificación
24         # Asks the user to enter a grade
25
26         if Val.ValidarNumeros(self.a):        # Verifica si la entrada es un número válido usando la clase Validacion
27             # Checks if the input is a valid number using the Validacion class
28             self.num += 1                      # Incrementa el contador de calificaciones
29             # Increments the grade counter
30             self.lista.append(int(self.a))    # Convierte la calificación a entero y la agrega a la lista
31             # Converts the grade to integer and appends it to the list
32
33         if self.num >= 5:                   # Si ya se ingresaron 5 calificaciones
34             # If 5 grades have been entered
35             print(self.lista)              # Muestra la lista de calificaciones
36             # Prints the list of grades
37             print(f'El promedio es {Val.Promedio(self.lista)})') # Calcula y muestra el promedio usando el método Promedio()
38             # Calculates and prints the average using the Promedio() method
39
40         else:                            # Si aún no se ingresan las 5 calificaciones
41             # If fewer than 5 grades have been entered
42             self.inicio()                # Llama de nuevo al método (recursividad)
43             # Calls the method again (recursion)
44
45         else:                            # Si la entrada no es un número
46             # If the input is not a valid number
47             print('No es un numero')      # Muestra un mensaje de error
48             # Displays an error message
49             self.inicio()                # Vuelve a pedir la entrada al usuario
50             # Calls the method again to ask for new input
51
52     if __name__ == '__main__':            # Comprueba si el archivo se está ejecutando directamente
53     # Checks if the file is being run directly
54     app = Principal()                 # Crea una instancia de la clase Principal
55     # Creates an instance of the Principal class
56     app.inicio()                     # Llama al método inicio() para iniciar el programa
57     # Calls the inicio() method to start the program
58

```

Ilustración 30 Código programa 2

```

PS C:\Users\oscar\Dropbox\UNIVERSIDAD\semestre\Estructura de datos\Segundo
Escribe una calificación
10
Escribe una calificación
5
Escribe una calificación
10
Escribe una calificación
8
Escribe una calificación
8
[10, 5, 10, 8, 8]
El promedio es 8.2
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\semestre\Estructura de datos\Segundo

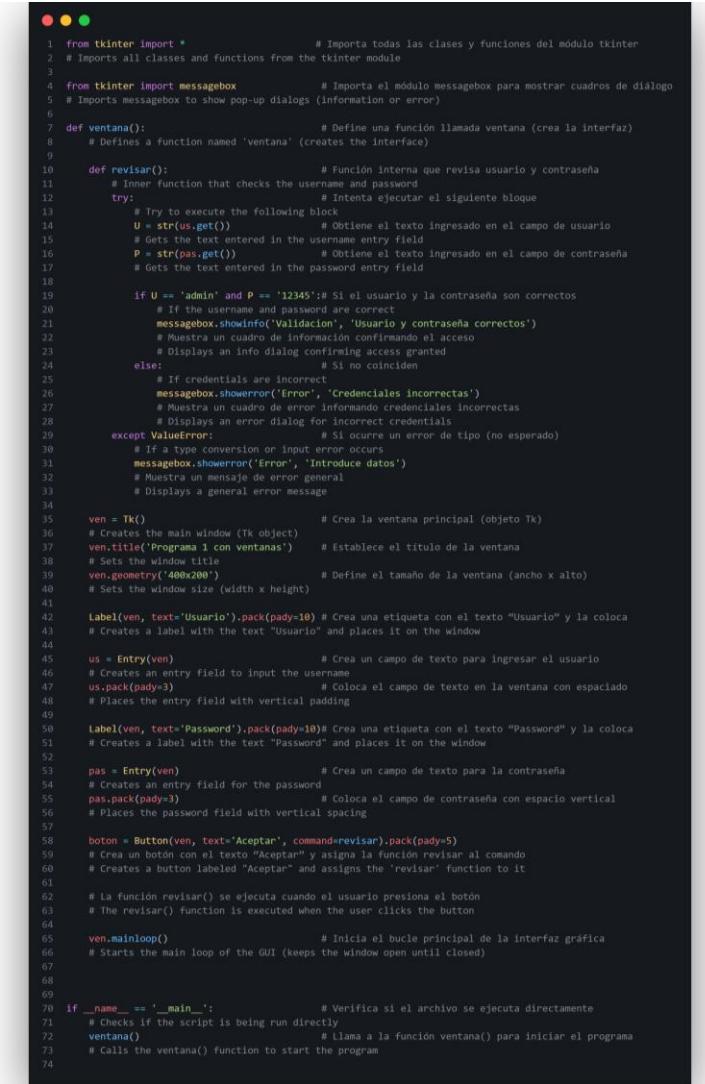
```

Ilustración 31 Ejecución programa2



Programa 3 inicio de sesión con ventanas

Este programa utiliza la biblioteca tkinter para generar ventanas , para crear una vista de inicio de sesión haciendo uso de etiquetas, cajas de texto y botones.



```

1  from tkinter import *           # Importa todas las clases y funciones del módulo tkinter
2  # Imports all classes and Functions from the tkinter module
3
4  from tkinter import messagebox   # Importa el módulo messagebox para mostrar cuadros de diálogo
5  # Imports messagebox to show pop-up dialogs (information or error)
6
7  def ventana():
8      # Define a function named 'ventana' (creates the interface)
9
10     def revisar():
11         # Inner function that checks the username and password
12         try:
13             # Try to execute the following block
14             U = str(us.get())           # Obtiene el texto ingresado en el campo de usuario
15             # Gets the text entered in the username entry field
16             P = str(pas.get())          # Obtiene el texto ingresado en el campo de contraseña
17             # Gets the text entered in the password entry field
18
19             if U == 'admin' and P == '12345':# Si el usuario y la contraseña son correctos
20                 # If the username and password are correct
21                 messagebox.showinfo('Validacion', 'Usuario y contraseña correctos')
22                 # Muestra un cuadro de información confirmando el acceso
23                 # Displays an info dialog confirming access granted
24             else:                      # Si no coinciden
25                 # If credentials are incorrect
26                 messagebox.showerror('Error', 'Credenciales incorrectas')
27                 # Muestra un cuadro de error informando credenciales incorrectas
28                 # Displays an error dialog for incorrect credentials
29             except ValueError:          # Si ocurre un error de tipo (no esperado)
30                 # If a type conversion or input error occurs
31                 messagebox.showerror('Error', 'Introduce datos')
32                 # Muestra un mensaje de error general
33                 # Displays a general error message
34
35             ven = Tk()                  # Crea la ventana principal (objeto Tk)
36             # Creates the main window (Tk object)
37             ven.title('Programa 1 con ventanas') # Establece el título de la ventana
38             # Sets the window title
39             ven.geometry('400x200')        # Define el tamaño de la ventana (ancho x alto)
40             # Sets the window size (width x height)
41
42             Label(ven, text='Usuario').pack(pady=10) # Crea una etiqueta con el texto "Usuario" y la coloca
43             # Creates a label with the text "Usuario" and places it on the window
44
45             us = Entry(ven)                # Crea un campo de texto para ingresar el usuario
46             # Creates an entry field to input the username
47             us.pack(pady=3)              # Coloca el campo de texto en la ventana con espacio vertical
48             # Places the entry field with vertical padding
49
50             Label(ven, text='Password').pack(pady=10) # Crea una etiqueta con el texto "Password" y la coloca
51             # Creates a label with the text "Password" and places it on the window
52
53             pas = Entry(ven)                # Crea un campo de texto para la contraseña
54             # Creates an entry field for the password
55             pas.pack(pady=3)              # Coloca el campo de contraseña con espacio vertical
56             # Places the password field with vertical padding
57
58             boton = Button(ven, text='Aceptar', command=revisar).pack(pady=5)
59             # Crea un botón con el texto "Aceptar" y asigna la función revisar al comando
60             # Creates a button labeled "Aceptar" and assigns the "revisar" function to it
61
62             # La función revisar() se ejecuta cuando el usuario presiona el botón
63             # The revisar() function is executed when the user clicks the button
64
65             ven.mainloop()                # Inicia el bucle principal de la interfaz gráfica
66             # Starts the main loop of the GUI (keeps the window open until closed)
67
68
69
70         if __name__ == '__main__':       # Verifica si el archivo se ejecuta directamente
71             # Checks if the script is being run directly
72             ventana()                  # llama a la función ventana() para iniciar el programa
73             # Calls the ventana() function to start the program
74

```

Ilustración 32 Código programa 3

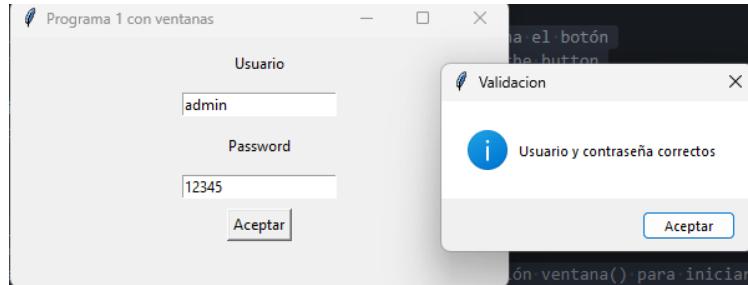


Ilustración 33 Ejecución programa 3



Programa 4 inicio de sesión con ventanas y PACK

Este programa utiliza la biblioteca tkinter para generar ventanas ,haciendo uso de clases y funciones para crear una vista de inicio de sesión haciendo uso de etiquetas, cajas de texto y botones.

```

1  from tkinter import *                                # Importa todos los componentes del módulo tkinter
2  # Imports all components From the tkinter module
3
4  from tkinter import messagebox                         # Importa messagebox para mostrar mensajes emergentes
5  # Imports messagebox to display pop-up messages (Info/error)
6
7
8  class ventana():
9      # Defines a class called 'ventana'
10
11     def __init__(self):                               # Método constructor: se ejecuta al crear el objeto
12         # When this method runs automatically when the object is created
13         self.ven = Tk()                                # Crea el objeto principal de la ventana
14         # Creates the main window object
15         self.ven.title('Programa 1 con ventanas')# Establece el título de la ventana
16         # Sets the window title
17         self.ven.geometry('400x200')                  # Define el tamaño de la ventana (ancho x alto)
18         # Sets the window size (width x height)
19         self.inicio()                                 # Llama al método inicio() para crear los widgets
20         # Calls the inicio() method to create the widgets
21
22     def inicio(self):                               # Método que construye la interfaz gráfica
23         # Method that builds the graphical interface
24         Label(self.ven, text='Usuario').pack(pady=3)   # Crea una etiqueta con el texto 'Usuario' y la agrega a la ventana
25         # Creates a label with the text 'Usuario' and adds it to the window
26
27         self.us = Entry(self.ven)                      # Crea un campo de texto para el nombre de usuario
28         # Creates a text entry field for the username
29         self.us.pack(pady=3)                          # Muestra el campo con espacio vertical
30         # Displays the entry field with vertical padding
31
32         Label(self.ven, text='Password').pack(pady=18) # Crea una etiqueta con el texto 'Password' y la agrega a la ventana
33         # Creates a text entry field for the password
34         self.pas.pack(pady=3)                         # Muestra el campo de contraseña
35         # Displays the password entry field
36
37         self.pas = Entry(self.ven)                      # Crea un campo de texto para la contraseña
38         # Creates a text entry field for the password
39         self.pas.pack(pady=3)                          # Muestra el campo de contraseña
40         # Displays the password entry field
41
42         boton = Button(self.ven, text='Aceptar', command=self.revisar).pack(pady=5) # Crea un botón con texto 'Aceptar' que ejecuta el método revisar() al hacer clic
43         # Creates a button labeled 'Aceptar' that calls the revisar() method when clicked
44
45
46         self.ven.mainloop()                           # Inicia el bucle principal de la interfaz gráfica
47         # Starts the main event loop of the graphical interface
48
49     def revisar(self):                            # Método que valida el usuario y la contraseña
50         # Method that validates the username and password
51         try:                                     # Bloque try para capturar posibles errores
52             # Try block to catch possible input errors
53             U = str(self.us.get())                # Obtiene el texto del campo de usuario
54             # Gets the text from the username entry field
55             P = str(self.pas.get())                # Obtiene el texto del campo de contraseña
56             # Gets the text from the password entry field
57
58             if U == 'admin' and P == '12345':       # Comprueba si usuario y contraseña son correctos
59                 # Checks if username and password are correct
60                 messagebox.showinfo('Validacion', 'Usuario y contraseña correctos') # Muestra un mensaje de validación exitosa
61                 # Muestra un mensaje de validación exitosa
62                 # Displays a success message
63             else:                                    # Si las credenciales son incorrectas
64                 # If credentials are incorrect
65                 messagebox.showerror('Error', 'Credenciales incorrectas') # Muestra un mensaje de error
66                 # Muestra un mensaje de error
67                 # Displays an error message
68
69         except ValueError:                       # Si ocurre un error inesperado de tipo
70             # If an unexpected type error occurs
71             messagebox.showerror('Error', 'Introduce datos') # Muestra un mensaje de error general
72             # Muestra un mensaje de error general
73             # Displays a general error message
74
75     if __name__ == '__main__':                  # Verifica si el script se ejecuta directamente
76         # Checks if the script is being run directly
77         app = ventana()                        # Crea una instancia de la clase ventana
78         # Creates an instance of the 'ventana' class
79

```

Ilustración 34 Código programa 4

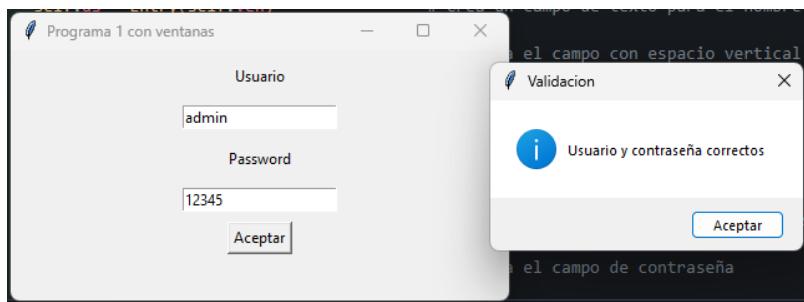


Ilustración 35 Ejecución programa 4



Programa 5 promedio de números con PLACE

Este programa usa cajas de texto para ingresar números para agregarlos a una lista que se muestra en la ventana del programa generando un promedio de los datos ingresados y un promedio general del total de datos

```

1  from tkinter import *           # Importa todos los componentes del módulo tkinter
2  # Imports all components from the tkinter module
3
4  from tkinter import messagebox   # Importa messagebox para mostrar alertas y errores
5  # Imports messagebox to show alert and error pop-ups
6
7
8  class principal():
9      # Defines a class called 'principal'
10     def __init__(self):          # Método constructor: se ejecuta al crear el objeto
11         # Constructor method: runs when the object is created
12         self.ven = Tk()           # Crea la ventana principal
13         self.ven.title('Programa 5 con ventanas')  # Establece el título de la ventana
14         self.ven.geometry('400x250')    # Define el tamaño de la ventana (ancho x alto)
15         self.lista = []            # Crea una lista para guardar los promedios individuales
16         self.suma=0                 # Crea un list para store individual average
17         self.promediar()          # Llama al método inicio() para construir la interfaz
18         self.inicio()              # Calls the inicio() method to build the interface
19
20     def suma(self):
21         # Método que sum all the values in the lista
22         # Suma todos los valores en la lista
23         s = 0                     # Inicializa la variable acumuladora
24         # Initializes the sum accumulator
25         for i in self.lista:        # Recorre cada elemento de la lista
26             # Iterates through each element in the list
27             s += i                  # Suma cada valor a la variable acumuladora
28             # Adds each value to the accumulator
29             # Returns s                # Retorna la suma total
30
31     def promediar(self):
32         # Método para calcular y mostrar el promedio
33         # Try block to handle conversion errors
34         try:
35             # Try block to handle conversion errors
36             a = float(self.n1.get())          # Convierte el valor del primer campo a número
37             # Converts first entry value to float
38             b = float(self.n2.get())          # Convierte el valor del segundo campo a número
39             # Converts second entry value to float
40             c = float(self.n3.get())          # Convierte el valor del tercer campo a número
41             # Converts third entry value to float
42             d = float(self.n4.get())          # Convierte el valor del cuarto campo a número
43             # Converts fourth entry value to float
44
45             pro = (a + b + c + d) / 4        # Calcula el promedio de los cuatro valores
46             # Calculates the average of the four numbers
47             self.l10.config(text=str(pro))    # Muestra el promedio en la etiqueta 16
48             # Displays the average in label 16
49
50             self.lista.append(pro)           # Agrega el promedio a la lista general
51             # Appends the average to the general list
52             self.l17.config(text=str(self.lista))  # Muestra la lista actualizada en la interfaz
53             # Updates and displays the list on the interface
54
55             # Limpia los campos de entrada después del cálculo
56             # Clears all entry fields after calculation
57             self.n1.delete(0, END)
58             self.n2.delete(0, END)
59             self.n3.delete(0, END)
60             self.n4.delete(0, END)
61
62             suma = self.suma()              # Llama al método suma() para obtener el total
63             # Calls suma() method to get total of averages
64             p = suma / len(self.lista)      # Calcula el promedio general de todos los promedios
65             # Calculates overall average of all averages
66             self.l11.config(text="Promedio general: " + str(p))  # Muestra el promedio general en la etiqueta 18
67             # Displays the general average in label 18
68
69         except ValueError:                 # Si ocurre un error al convertir (entrada no numérica)
70             # If a conversion error occurs (non-numeric input)
71             messagebox.showerror("Error", "Algun dato no es número")  # Muestra un error pop-up message
72             # Displays an error pop-up message
73             self.n1.delete(0, END)          # Limpiar el campo que el usuario valvía a escribir
74             # Clears entry fields so user can re-enter data
75             self.n2.delete(0, END)
76             self.n3.delete(0, END)
77             self.n4.delete(0, END)
78
79     def salir(self):                    # Método para cerrar la ventana
80         # Método para cerrar la ventana
81         self.ven.destroy()             # Destroys (closes) the main window
82
83     def inicio(self):                  # Método que construye los componentes de la GUI
84         # Method that builds the GUI components
85         # Las coordenadas y corresponden a filas, x a columnas
86         # y coordinates = rows, x coordinates = columns
87
88         self.n1 = Entry(self.ven)      # Campo de entrada 1
89         self.n1.place(y=10, x=140)    # Campo de entrada 1
90         self.n2 = Entry(self.ven)      # Campo de entrada 2
91         self.n2.place(y=50, x=140)    # Campo de entrada 2
92         self.n3 = Entry(self.ven)      # Campo de entrada 3
93         self.n3.place(y=90, x=140)    # Campo de entrada 3
94         self.n4 = Entry(self.ven)      # Campo de entrada 4
95         self.n4.place(y=130, x=140)    # Campo de entrada 4
96
97         self.l1 = Label(self.ven, text="Escribe un número").place(y=10, x=20)  # Etiqueta para el primer número
98         self.l2 = Label(self.ven, text="Escribe un número").place(y=50, x=20)  # Etiqueta para el segundo número
99         self.l3 = Label(self.ven, text="Escribe un número").place(y=90, x=20)  # Etiqueta para el tercer número
100        self.l4 = Label(self.ven, text="Escribe un número").place(y=130, x=20)  # Etiqueta para el cuarto número
101
102        self.b1 = Button(self.ven, text="Promedio").place(y=180, x=10)  # Botón que calcula el promedio / button that calculates the average
103        self.b2 = Button(self.ven, text="Salir", command=self.salir).place(y=90, x=300)  # Botón para cerrar el programa / button to exit the program
104
105        self.l17 = Label(self.ven, text="")  # Etiqueta vacía donde se mostrará el promedio
106        self.l16.place(y=180, x=190)
107
108        self.l11 = Button(self.ven, text="Promedio", command=self.promediar).place(y=50, x=300)  # Botón que calcula el promedio / button that calculates the average
109
110        self.l17 = Label(self.ven, text="Promedio general: 0.0").place(y=150, x=300)  # Etiqueta inicial del promedio general
111        self.l18 = Label(self.ven, text="").place(y=180, x=300)  # Label showing the initial overall average
112
113        self.ven.mainloop()  # Mantiene la ventana abierta (bucle principal)
114        # Keeps the window open (main event loop)
115
116
117        if __name__ == '__main__':
118            # Verifica si se ejecuta directamente el script
119            # Checks if the script is being run directly
120            app = principal()  # Crea un objeto de la clase principal
121
122            # Creates an instance of the 'principal' class
123
```

Ilustración 36 Código programa 5

Ilustración 37 Ejecución programa 5



Programa 6 ventanas con GRID

Este programa hace uso de una listview para mostrar visualmente los datos agregados por el usuario, con el uso de botones para agregar nuevas entradas, calcular el numero mayor de la lista y el numero menor de la lista.

```
1 // From tkinter import *           # Imports all components from the tkinter module
2 # Imports all components from the tkinter module
3 # Imports the messagebox module    # Imports messages para mostrar mensajes emergentes
4 # Imports messagebox to show pop-up messages (info or error)
5
6 class Principal():              # Define a class called "Principal"
7     def __init__(self):          # Method constructor; se ejecuta al crear el objeto
8         self.root = Tk()          # Create the window (Tk object)
9         self.root.title("Primer app con ventanas GUI") # Establece el título de la ventana
10        self.root.geometry("450x300") # Establece el tamaño de la ventana
11        self.root.wmiconify()      # Define el tamaño de la ventana (ancho x alto)
12        self.root.wmresizable(0,0)   # No permite que se redimensione la ventana
13        self.root.wmtitle("Principal") # Define el nombre de la ventana
14        self.root.wmheight(300)     # Define el tamaño de la ventana (alto x ancho)
15        self.root.wmwidth(450)      # Define el tamaño de la ventana (ancho x alto)
16        self.root.wmiconbitmap("") # No muestra icono en la barra de tareas
17        self.root.wmtitle("Principal") # Define el nombre de la ventana
18        self.root.wmheight(300)     # Define el tamaño de la ventana (alto x ancho)
19        self.root.wmwidth(450)      # Define el tamaño de la ventana (ancho x alto)
20        self.root.wmiconbitmap("") # No muestra icono en la barra de tareas
21        self.root.wmtitle("Principal") # Define el nombre de la ventana
22        self.root.wmheight(300)     # Define el tamaño de la ventana (alto x ancho)
23        self.root.wmwidth(450)      # Define el tamaño de la ventana (ancho x alto)
24        self.root.wmiconbitmap("") # No muestra icono en la barra de tareas
25
26        self.root.bind_all("Return", self.onEnter) # Llama a la función onEnter cuando se presione la tecla Enter
27
28        self.root.mainloop() # Inicia el bucle principal de la aplicación
29
30
31    def onEnter(self):             # Método que se ejecuta cuando se presiona la tecla Enter
32        self.var1.set(str(self.var1.get() + 1)) # Aumenta en 1 el valor del primer número ingresado
33        self.var2.set(str(self.var2.get() + 1)) # Aumenta en 1 el valor del segundo número ingresado
34
35        self.label1.config(text="{} + {} = {}".format(self.var1.get(), self.var2.get(), self.var1.get() + self.var2.get())) # Actualiza la etiqueta con el resultado
36
37
38    def onEnter2(self):            # Método que construye la interfaz gráfica
39        self.root.title("Calculadora") # Título de la aplicación
40        self.root.iconbitmap("calculadora.ico") # Icono de la aplicación
41        self.root.wmtitle("Principal") # Etiqueta del título principal
42
43        self.root.grid() # Establece la red de cuadros
44
45        self.label1 = Label(self.root, text="{} + {} = {}".format(self.var1.get(), self.var2.get(), self.var1.get() + self.var2.get()), font="Times New Roman, 20") # Etiqueta que muestra el resultado
46        self.label1.grid(row=0, column=0) # Coloca la etiqueta en la fila 0, columna 0
47
48        self.entry1 = Entry(self.root, width=10) # Campo de entrada para el primer número
49        self.entry1.grid(row=1, column=0) # Coloca el campo de entrada en la fila 1, columna 0
50
51        self.entry2 = Entry(self.root, width=10) # Campo de entrada para el segundo número
52        self.entry2.grid(row=1, column=1) # Coloca el campo de entrada en la fila 1, columna 1
53
54        self.button1 = Button(self.root, text="Sumar", command=self.sumar) # Botón para sumar los números
55        self.button1.grid(row=2, column=0) # Coloca el botón en la fila 2, columna 0
56
57        self.button2 = Button(self.root, text="Restar", command=self.restar) # Botón para restar los números
58        self.button2.grid(row=2, column=1) # Coloca el botón en la fila 2, columna 1
59
60        self.button3 = Button(self.root, text="Multiplicar", command=self.multiplicar) # Botón para multiplicar los números
61        self.button3.grid(row=2, column=2) # Coloca el botón en la fila 2, columna 2
62
63        self.button4 = Button(self.root, text="Dividir", command=self.dividir) # Botón para dividir los números
64        self.button4.grid(row=2, column=3) # Coloca el botón en la fila 2, columna 3
65
66        self.button5 = Button(self.root, text="Nuevo", command=self.nuevo) # Botón para iniciar una nueva operación
67        self.button5.grid(row=3, column=0) # Coloca el botón en la fila 3, columna 0
68
69        self.button6 = Button(self.root, text="Borrar", command=self.borrar) # Botón para borrar la lista
70        self.button6.grid(row=3, column=1) # Coloca el botón en la fila 3, columna 1
71
72        self.button7 = Button(self.root, text="Añadir", command=self.agregar) # Botón para agregar los números a la lista
73        self.button7.grid(row=3, column=2) # Coloca el botón en la fila 3, columna 2
74
75        self.button8 = Button(self.root, text="Mostrar", command=self.mostrar) # Botón para mostrar los números de la lista
76        self.button8.grid(row=3, column=3) # Coloca el botón en la fila 3, columna 3
77
78
79    def nuevo(self):               # Método para borrar la lista
80        self.listbox.delete(0, END) # Borra la lista
81
82        self.var1.set("") # Limpia el primer campo de texto
83        self.var2.set("") # Limpia el segundo campo de texto
84
85        self.label1.config(text="") # Limpia la etiqueta
86
87
88    def agregar(self):             # Método para agregar los números a la lista
89        self.listbox.insert(END, self.var1.get() + self.var2.get()) # Inserta el resultado en la lista
90
91        self.var1.set("") # Limpia el primer campo de texto
92        self.var2.set("") # Limpia el segundo campo de texto
93
94        self.label1.config(text="{} + {} = {}".format(self.var1.get(), self.var2.get(), self.var1.get() + self.var2.get())) # Actualiza la etiqueta con el resultado
95
96
97    def mostrar(self):             # Método para mostrar los números de la lista
98        self.listbox.config(selectmode="multiple") # Permite seleccionar más de un elemento
99        self.listbox.selection_set(0) # Selecciona el primer elemento
100       self.listbox.activate(0) # Activa el primer elemento
101
102       self.listbox.bind("Return", self.onEnter) # Etiqueta que mostrará la lista en pantalla
103
104       self.listbox.bind("Return", self.onEnter) # Bucle para correr el programa
105
106
107    def onEnter(self):             # Método para mostrar los números agregados visualmente
108        self.listbox.insert(END, self.var1.get() + self.var2.get(), column=0) # Inserta el resultado en la lista
109
110        self.listbox.config(selectmode="multiple") # Permite seleccionar más de un elemento
111        self.listbox.selection_set(0) # Selecciona el primer elemento
112
113        self.listbox.bind("Return", self.onEnter) # Etiqueta que mostrará la lista en pantalla
114
115        self.listbox.bind("Return", self.onEnter) # Bucle para correr el programa
116
117
118    def salirm(self):              # Método para cerrar la ventana
119        self.root.destroy() # Destruye la ventana principal
120
121
122    def iniciar(self):             # Método para iniciar la ventana
123        self.root.title("Principal") # Crea una instancia de la clase Principal
124        self.root.mainloop() # Llama al método iniciar() para iniciar la ventana
```

Ilustración 38 Código programa 6

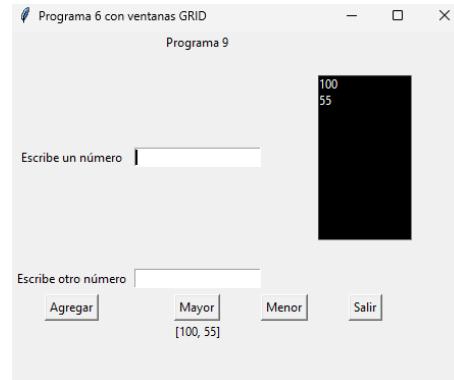


Ilustración 39 Ejecución programa 6



Programa 7 Uso de la biblioteca random

Este programa usa el programa anterior implementando la biblioteca random para agregar valores al azar e imprimiéndolo la lista en la consola

Ilustración 40 Código programa 7

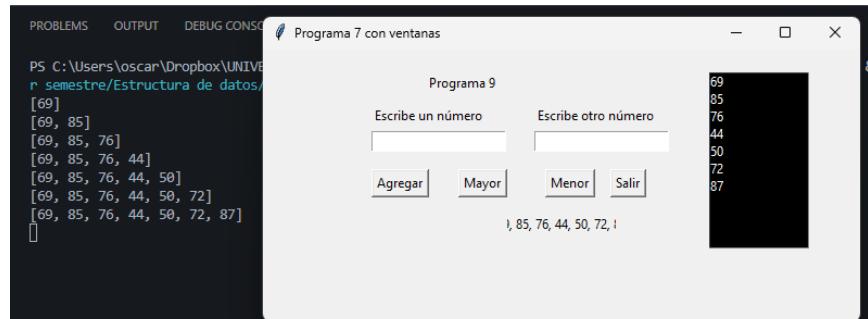


Ilustración 41 Ejecución programa 7

Tarea 1 ordenamiento de calificaciones messagebox

Este programa pide al usuario un nombre y 3 calificaciones, ordenándolas de mayor a menor y agregándolas a una lista, el programa termina hasta que el usuario no desee agregar más valores mostrando con messagebox los datos guardados en la lista.

```

 1  from tkinter import *
 2  from tkinter import messagebox
 3
 4  class inicio():
 5      def __init__(self):
 6          # EN: Create main window object
 7          # ES: Crear el objeto de la ventana principal
 8          self.ven = Tk()
 9          # EN: Set the window title
10          # ES: Establecer el título de la ventana
11          self.ven.title('Tarea con ventanas')
12          # EN: Set the window size (width x height)
13          # ES: Establecer el tamaño de la ventana (ancho x alto)
14          self.ven.geometry('300x100')
15          # EN: Call the method to create the window widgets
16          # ES: Llamar al método para crear los widgets de la ventana
17          self.ven.mainloop()
18
19
20  def ventana(self):
21      # EN: Add a label and entry for student name
22      # ES: Agregar una etiqueta y entrada para el nombre del alumno
23      label(self.ven,text='Nombre alumno').pack(pady=10)
24      self.nom = Entry(self.ven)
25      self.nom.pack(pady=3)
26
27      # EN: Add a label and entry for grade 1
28      # ES: Agregar una etiqueta y entrada para la calificación 1
29      label(self.ven,text='Calificación 1').pack(pady=10)
30      self.c1 = Entry(self.ven)
31      self.c1.pack(pady=3)
32
33      # EN: Add a label and entry for grade 2
34      # ES: Agregar una etiqueta y entrada para la calificación 2
35      label(self.ven,text='Calificación 2').pack(pady=10)
36      self.c2 = Entry(self.ven)
37      self.c2.pack(pady=3)
38
39      # EN: Add a label and entry for grade 3
40      # ES: Agregar una etiqueta y entrada para la calificación 3
41      label(self.ven,text='Calificación 3').pack(pady=10)
42      self.c3 = Entry(self.ven)
43      self.c3.pack(pady=3)
44
45      # EN: Add a button to save data and call the revisar method
46      # ES: Agregar un botón para guardar los datos y llamar al método revisar
47      boton = Button(self.ven,text='Guardar',command=self.revisar).pack(pady=5)
48
49      # EN: Start the Tkinter event loop
50      # ES: Iniciar el bucle de eventos de Tkinter
51      self.ven.mainloop()
52
53  def revisar(self):
54      # EN: Create a temporary list to store name and grades
55      # ES: Crear una lista temporal para almacenar nombre y calificaciones
56      listal=[]
57
58      # EN: Get data from entry widgets
59      # ES: Obtener los datos de los widgets de entrada
60      nom = str(self.nom.get())
61      c1= int(self.c1.get())
62      c2= int(self.c2.get())
63      c3= int(self.c3.get())
64
65      # EN: Compare grades to determine highest, middle, and lowest
66      # ES: Comparar calificaciones para determinar mayor, medio y menor
67      if (c1 > c2):
68          if (c1>c3):
69              messagebox.showinfo('informacion','el mayor es(c1)')
70              listal.append(nom)
71              listal.append(c1)
72              messagebox.showinfo('Informacion','el de en medio es (c2)')
73
74      else:
75          if (c2>c3):
76              messagebox.showinfo('informacion','el mayor es(c2)')
77              listal.append(nom)
78              listal.append(c2)
79              messagebox.showinfo('Informacion','el menor es(c3)')
80
81      else:
82          messagebox.showinfo('informacion','el de en medio es(c3)')
83          listal.append(nom)
84          listal.append(c3)
85          messagebox.showinfo('Informacion','el menor es(c1)')
86
87      # EN: Ask if the user wants to add another student
88      # ES: Preguntar si el usuario desea agregar otro alumno
89      resp=messagebox.askyesno('Informacion','deseas agregar otro alumno')
90      if resp:
91          # EN: If yes, do nothing (the window stays open)
92          # ES: Si es si, no hacer nada (la ventana permanece abierta)
93          pass
94      else:
95          # EN: Show the final list and exit
96          # ES: Mostrar la lista final y salir
97          messagebox.showinfo('Informacion',listal)
98          exit()
99
100 # EN: Global list to store all students and their grades
101 # ES: Lista global para almacenar todos los alumnos y sus calificaciones
102 lista=[]
103
104 if __name__ == "__main__":
105     # EN: Start the GUI application
106     # ES: Iniciar la aplicación de la GUI
107     app=inicio()

```

Ilustración 42 Código Tarea 1

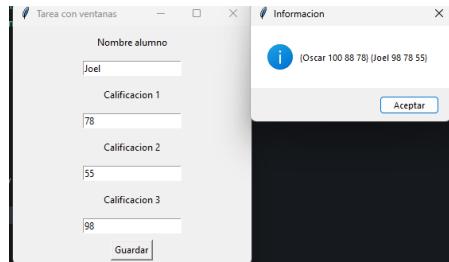


Ilustración 43 Ejecución tarea 1



Tarea 1.1 ordenamiento de calificaciones consola

Este programa pide al usuario un nombre y 3 calificaciones, ordenándolas de mayor a menor y agregándolas a una lista, el programa termina hasta que el usuario no desee agregar más valores mostrando en consola los datos guardados en la lista.

Ilustración 44 Código tarea 1.1

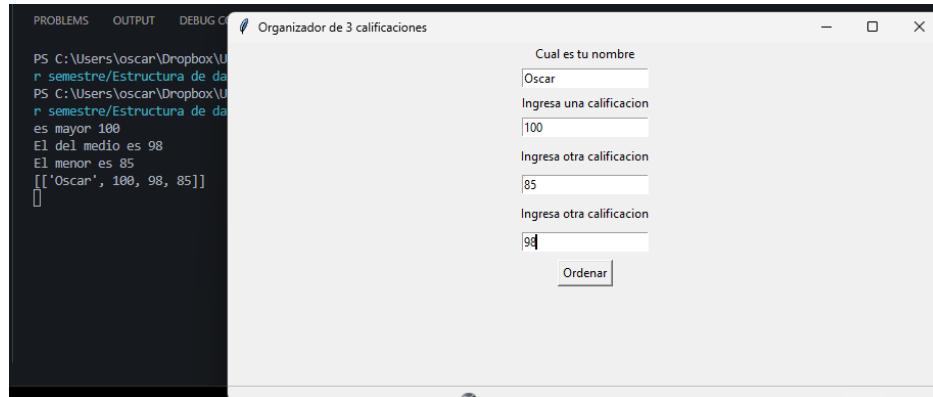


Ilustración 45 Ejecución tarea 1.1



solución examen unidad 1

opción de solución al examen aplicado en la unidad 1 haciendo un ordenamiento de mayor a menor de 3 calificaciones ingresadas por el usuario

```

1 def inicio():
2     # EN: Create a temporary list to store the name and grades
3     # ES: Crear una lista temporal para almacenar el nombre y las calificaciones
4     lista1=[]
5
6     # EN: Ask the user for a name
7     # ES: Pedir al usuario un nombre
8     nom = input('Escribe un nombre')
9
10    # EN: Ask the user for three grades (converted to integers)
11    # ES: Pedir al usuario tres calificaciones (convertidas a enteros)
12    c1= int(input('Escribe una calificación'))
13    c2= int(input('Escribe una calificación'))
14    c3= int(input('Escribe una calificación'))
15
16    # EN: Compare grades to determine the order: highest, middle, lowest
17    # ES: Comparar calificaciones para determinar el orden: mayor, medio, menor
18    if (c1 > c2):
19        if (c1>c3):
20            # EN: c1 is the highest
21            # ES: c1 es la mayor
22            print('es mayor {c1}')
23            lista1.append(nom)
24            lista1.append(c1)
25        if (c2>c3):
26            # EN: c2 is middle, c3 is lowest
27            # ES: c2 es el del medio, c3 es la menor
28            print('es el de en medio {c2}')
29            print('es el menor {c3}')
30            lista1.append(c2)
31            lista1.append(c3)
32            lista2.append(lista1)
33        else:
34            # EN: c3 is middle, c2 is lowest
35            # ES: c3 es el del medio, c2 es la menor
36            print('es el de en medio {c3}')
37            print('es el menor {c2}')
38            lista1.append(c3)
39            lista1.append(c2)
40            lista2.append(lista1)
41    else:
42        # EN: c3 is highest, c1 is middle, c2 is lowest
43        # ES: c3 es la mayor, c1 es el del medio, c2 es la menor
44        print('es el de en medio {c1}')
45        print('es el mayor {c3}')
46        print('es el menor {c2}')
47        lista1.append(nom)
48        lista1.append(c3)
49        lista1.append(c1)
50        lista1.append(c2)
51        lista2.append(lista1)
52
53    else:
54        if (c2>c3):
55            # EN: c2 is highest
56            # ES: c2 es la mayor
57            print('es el mayor {c2}')
58            lista1.append(nom)
59            lista1.append(c2)
60        if (c1>c3):
61            # EN: c1 is middle, c3 is lowest
62            # ES: c1 es el del medio, c3 es la menor
63            print('es el de en medio {c1}')
64            print('es el menor {c3}')
65            lista1.append(c1)
66            lista1.append(c3)
67            lista2.append(lista1)
68
69    else:
70        # EN: c3 is highest, c1 is middle, c2 is lowest
71        # ES: c3 es la mayor, c1 es el del medio, c2 es la menor
72        print('es el de en medio {c3}')
73        print('es el mayor {c1}')
74        print('es el menor {c2}')
75        lista1.append(c3)
76        lista1.append(c1)
77        lista1.append(c2)
78        lista2.append(lista1)
79
80    # EN: Global list to store all persons and their ordered grades
81    # ES: Lista global para almacenar todas las personas y sus calificaciones ordenadas
82    lista2=[]
83
84    if __name__ == "__main__":
85        # EN: Start the program and ask for additional persons
86        # ES: Iniciar el programa y preguntar por más personas
87        inicio()
88        while (True):
89            # EN: Ask if the user wants to enter another person
90            # ES: Preguntar si el usuario desea ingresar otra persona
91            a = input('Deseas otra persona')
92            if a=="N" or a=="n" or a=="NO" or a=="no":
93                # EN: Print the final list and exit loop
94                # ES: Imprimir la lista final y salir del bucle
95                print(lista2)
96                break

```

Ilustración 46 Código solución de examen

```
PS C:\Users\oscar\Dropbox\UNIVERSIDAD\Tercer semestre\Estructura de  
r semestre\Estructura de datos\Segundo Parcial\Practicas unidad 2\sc  
Escribe un nombreoscar  
Escribe hna calificacion100  
Escribe hna calificacion87  
Escribe hna calificacion98  
es mayor 100  
es el de en medio 98  
es el menor 87  
Deseas otra persona
```

Ilustración 47 Ejecución solución de examen

Validaciones de números y cálculo de promedio

Validaciones usadas en diferentes programas para validar si el dato son números y calcular un promedio.

```
● ● ●

1  class Validacion():
2      def __init__(self):
3          # EN: Initialize sum and average attributes
4          # ES: Inicializar los atributos de suma y promedio
5          self.suma = 0
6          self.promedio = 0.0
7          pass
8
9      def ValidarNumeros(self,valor):
10         # EN: Check if the input value is a number (all digits)
11         # ES: Verificar si el valor de entrada es un número (solo dígitos)
12         if valor.isdigit():
13             return True
14         else:
15             return False
16
17     def Promedio(self,lista):
18         # EN: Calculate the sum of all elements in the list
19         # ES: Calcular la suma de todos los elementos de la lista
20         for i in lista:
21             self.suma += i
22         # EN: Calculate the average
23         # ES: Calcular el promedio
24         self.promedio = self.suma / len(lista)
25
26         return self.promedio
```



Repasso

Repasso para examen de los temas visto en la segunda unidad

```

● ● ●
1  from tkinter import *
2  from tkinter import messagebox
3  from validarrepasso import validar
4
5  class Principal():
6      def __init__(self):
7          # EN: Create the main window
8          # ES: Crear la ventana principal
9          self.ventana = Tk()
10         # EN: Set the window size (width x height)
11         # ES: Establecer el tamaño de la ventana (ancho x alto)
12         self.ventana.geometry("400x200")
13         # EN: Initialize an empty list to store input data
14         # ES: Inicializar una lista vacía para almacenar los datos ingresados
15         self.lista=[]
16         # EN: Create an instance of the validation class
17         # ES: Crear una instancia de la clase de validación
18         self.valid = validar()
19
20     def inicio(self):
21         # EN: Add a label as the program title
22         # ES: Agregar un label como título al programa
23         Label(self.ventana, text="Programa de Python con TKInter").place(x=90,y=20)
24         # EN: Create a label to show the list of data
25         # ES: Crear un label que muestra la lista de datos
26         Label(self.ventana,text="Escribe un dato").place(x=50,y=50)
27         # EN: Create an entry widget to input data
28         # ES: Crear un widget de entrada para escribir datos
29         self.dato = Entry(self.ventana)
30         self.dato.place(x=150,y=50,width=150)
31         # EN: Create a button that calls the validation method
32         # ES: Crear un botón que llama al método de validación
33         Button(self.ventana, text="Validar", command=self.validarDatos).place(x=200,y=100,width=150)
34         # EN: Label to show the list of data
35         # ES: Label para mostrar la lista de datos
36         self.mostrar = Label(self.ventana, text="Ejemplo")
37         self.mostrar.place(x=20,y=150)
38
39         # EN: Start the Tkinter event loop
40         # ES: Iniciar el bucle de eventos de Tkinter
41         self.ventana.mainloop()
42
43     def validarDatos(self):
44         # EN: Get the value from the entry widget
45         # ES: Obtener el valor del widget de entrada
46         val = self.dato.get()
47         if val != "":
48             # EN: Add value to the list
49             # ES: Agregar el valor a la lista
50             self.lista.append(val)
51             # EN: Print the value for checking
52             # ES: Imprimir el valor para revisión
53             self.Revisar(val)
54             # EN: Clear the entry widget
55             # ES: Limpiar el widget de entrada
56             self.dato.delete(0,END)
57             # EN: Update the label to show the list of data
58             # ES: Actualizar el label para mostrar la lista de datos
59             self.mostrar.config(text=f'{self.lista}')
60             # EN: Validate the value using the validation class
61             # ES: Valida el valor usando la clase de validación
62             # respuesta = self.valid.ValidarAscii(val)
63             # respuesta = self.valid.ValidarConString(val)
64             # respuesta = self.valid.ValidarConString(val)
65             # EN: Show a message box with the validation result
66             # ES: Mostrar un mensaje con el resultado de la validación
67             messagebox.showinfo("Validar datos", f'El dato es: {respuesta}')
68
69         else:
70             # EN: Show an error message if entry is empty
71             # ES: Mostrar mensaje de error si la caja de texto está vacía
72             messagebox.showerror("Error", "Caja de texto esta vacía")
73
74     def Revisar(self,v):
75         # EN: Print the value in console (for debugging)
76         # ES: Imprimir el valor en consola (para depuración)
77         print(v)
78
79 if __name__ == '__main__':
80     # EN: Create an instance of Principal and start the GUI
81     # ES: Crear una instancia de Principal y ejecutar la GUI
82     app=Principal()
83     app.inicio()

```

Ilustración 48 Código repaso

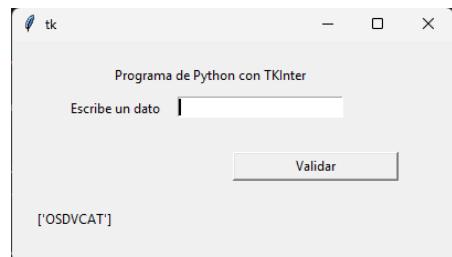


Ilustración 49 ejecución repaso



Validación repaso

Validaciones usadas en el repaso para validar si un dato es número, son letras o es un correo con diferentes métodos.

```

1  class validar():
2      def __init__(self):
3          # EN: Initialize an index attribute for string validation
4          # ES: Inicializar un atributo índice para la validación de cadenas
5          self.index = 0
6          pass
7
8      def ValidarAscii(self,valor):
9          # EN: Count digits and letters in the input string
10         # ES: Contar números y letras en la cadena de entrada
11         con = 0
12         con2 = 0
13         for i in valor:
14             # EN: Check if character is a digit (ASCII 48-57)
15             # ES: Verificar si el carácter es un número (ASCII 48-57)
16             if ord(i) >= 48 and ord(i) <= 57:
17                 con += 1
18             # EN: Check if character is a letter (uppercase or lowercase)
19             # ES: Verificar si el carácter es una letra (mayúscula o minúscula)
20             if (ord(i) >= 65 and ord(i) <= 90) or (ord(i) >= 97 and ord(i) <= 122):
21                 con2 += 1
22
23         # EN: Determine the type of input based on counts
24         # ES: Determinar el tipo de entrada según los conteos
25         if con == len(valor):
26             return "Numeros"
27         elif con2 == len(valor):
28             return "Letras"
29         else:
30             return "Letras y numeros"
31
32     def ValidarConError(self,valor):
33         # EN: Validate input using try-except for integers and floats
34         # ES: Validar la entrada usando try-except para enteros y flotantes
35         a = 0
36         b = 0.0
37         try:
38             a = int(valor)
39             return "Numeros"
40         except ValueError:
41             try:
42                 b = float(valor)
43                 return "Es numero real o con decimales"
44             except ValueError:
45                 return "Letras o numeros"
46
47     def ValidarConString(self,valor):
48         # EN: Recursively check if the string contains an '@' symbol
49         # ES: Revisar recursivamente si la cadena contiene un símbolo '@'
50
51         # EN: Base case: if index is within the string length
52         # ES: Caso base: si el índice está dentro de la longitud de la cadena
53         if self.index < len(valor):
54             if valor[self.index] == '@':
55                 return "Si es un correo"
56             else:
57                 if self.index < len(valor):
58                     self.index += 1
59                     # EN: Recursive call to continue searching
60                     # ES: Llamada recursiva para continuar buscando
61                     return self.ValidarConString(valor)
62                 else:
63                     return "No es un correo"
64             else:
65                 return "No es un correo"
66

```

Ilustración 50 Código validaciones repaso



Preexamen

Preexamen de la segunda unidad donde por medio de una ventana y cajas de texto valida si los datos ingresados son : letras minúsculas, letras mayúsculas y números para agregarlas a un a listview.

```

1 # Hacer un programa que mediante una ventana de tkinter realice lo siguiente:
2 # 1 - Una ventana principal con tres cajas de texto y tres botones
3 # 2 - Al pulsar el botón validar se imprime en la primera caja de texto por letra
4 # minúsculas, en la segunda caja solo letras mayúsculas y en la tercera caja
5 # numeros, dicha validación se realizara el primer botón, el segundo botón es
6 # para borrar los datos de las cajas de texto y el tercer botón para cerrar la ventana
7 # 3 - Se concatenaran y se agregarán a una lista visual (listbox) y en una
8 # cuarta caja de texto se muestra el número de elementos de la lista
9 # 4 - Se habilita un botón para cerrar la ventana de ejecución de la lista
10 from tkinter import *
11 from tkinter import messagebox
12 from Validation import validationexamen # EN: Import validation class / ES: Importar la clase de validacion
13
14 class Programa():
15     def __init__(self):
16         # EN: Create the main window / ES: Crear la ventana principal
17         self.root = Tk() # Tk()
18         self.root.title("Tk") # EN: Set window title / ES: Establecer titulo de ventana
19         self.root.geometry("550x300") # EN: Set window size / ES: Establecer tamano de ventana
20         self.listas = [] # EN: List to store data / ES: Lista para almacenar datos
21         self.validacion = validationexamen() # EN: Validation object / ES: Objeto de validacion almacenado
22         self.banderas = False # EN: Flag to control validation / ES: Bandera para controlar la validacion
23         self.valid = validationexamen() # EN: Validation object / ES: Objeto de validacion
24
25     def inicio(self):
26         # EN: Labels for titles and field names / ES: Etiquetas para titulos y nombres de campos
27         Label(self.root, text="Pre-Examen").place(x=150,y=10)
28         Label(self.root, text="Nombre").place(x=150,y=50)
29         Label(self.root, text="Mayusculas").place(x=150,y=50)
30         Label(self.root, text="Numeros").place(x=150,y=50)
31
32         # EN: Text boxes / ES: Cajas de texto
33         self.nombreEntry = self.root.Entry(self.root).place(x=150,y=80)
34         self.mayusEntry = self.root.Entry(self.root).place(x=150,y=80)
35         self.numerosEntry = self.root.Entry(self.root).place(x=150,y=80)
36         self.nombre.place(x=150,y=70,width=80)
37         self.mayus.place(x=150,y=70,width=80)
38         self.numeros.place(x=150,y=70,width=80)
39
40         # EN: Buttons to validate and add / ES: Botones para validar y agregar
41         self.valid.button(self.root, text="Validar", command=self.validar)
42         self.valid.place(x=150,y=150,width=100)
43         self.agregarbutton = self.root.Button(self.root, text="Añadir", command=self.agregar)
44         self.agregarbutton.place(x=150,y=150,width=100)
45
46         # EN: Label that shows the number of stored elements / ES: Etiqueta que muestra la cantidad de elementos almacenados
47         self.contadorLabel = self.root.Label(self.root, text="Datos almacenado en lista: 0")
48         self.contadorLabel.place(x=150,y=180)
49
50         # EN: Listbox to display the stored elements / ES: Listbox para mostrar los elementos almacenados
51         self.listbox = Listbox(self.root, width=10, height=10, bg="gray",
52                               activestyle="dotbox", fg="black")
53         self.listbox.place(x=150,y=200)
54
55         # EN: Open the window open / ES: Hacer la ventana abierta
56         self.root.mainloop()
57
58     def validar(self):
59         # EN: Get values from the Entry fields / ES: Obtener valores de las cajas de texto
60         self.nombre = self.root.Entry(self.root).get()
61         self.mayus = self.root.Entry(self.root).get()
62         self.numeros = self.root.Entry(self.root).get()
63
64         # EN: Check for empty fields / ES: Verificar si hay campos vacios
65         if self.nombre == "" or self.mayus == "" or self.numeros == "":
66             messagebox.showerror("Error", "Falta un dato")
67
68         else:
69             # EN: Validate uppercase / ES: Validar mayusculas
70             if self.valid.validarM(self.nombre) == True:
71                 messagebox.showerror("Error", "No son letras mayusculas")
72
73             # EN: Validate lowercase / ES: Validar minúsculas
74             if self.valid.validarM(self.mayus) == True:
75                 messagebox.showerror("Error", "No son letras minúsculas")
76
77             # EN: Validate numbers / ES: Validar numeros
78             if self.valid.validarN(self.numeros) == True:
79                 messagebox.showerror("Error", "No son numeros ")
80
81             # EN: If all fields are validated / ES: Si todos los campos son correctos
82             if self.valid.validarU(self.nombre) and self.valid.validarM(self.mayus) and self.valid.validarN(self.numeros) == True:
83                 messagebox.showinfo("Info", "Campos validados")
84                 self.banderas = True
85
86     def agregar(self):
87         # EN: Only add the fields are validated / ES: Solo agregar si los campos estan validados
88         if self.banderas == True:
89             valor = str(self.root.Entry(self.root).get())
90             self.listbox.insert(0, valor)
91             self.root.Entry(self.root).delete(0, "end")
92             self.listbox.size() + 1, valor
93             self.con = 1
94
95             # EN: Update counter label / ES: Actualizar la etiqueta del contador
96             self.contadorLabel.config(text="Datos almacenado en lista: "+str(self.con))
97             self.banderas = False
98
99             # EN: Clear the text boxes / ES: Limpiar las cajas de texto
100            self.nombre.delete(0,END)
101            self.mayus.delete(0,END)
102            self.numeros.delete(0,END)
103            self.listbox.delete(0,END)
104
105        else:
106            # EN: Error if fields are not validated / ES: Error si los campos no estan validados
107            messagebox.showerror("Error", "No esta valido")
108
109
110     # EN: Program entry point / ES: Punto de entrada del programa
111     if __name__ == "__main__":
112         app = Programa()
113         app.mainloop()
114
115 
```

Ilustración 51 Código preexamen

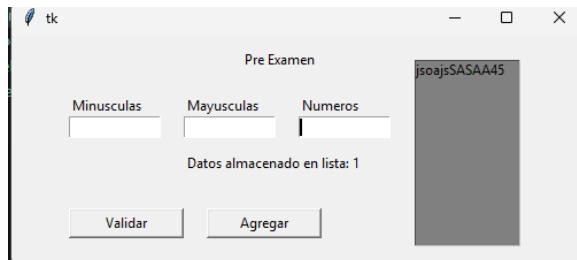


Ilustración 52 Ejecución preexamen

Validaciones preexamen

Validaciones utilizadas en el preexamen con uso de la recursividad

```

1
2 class validacionexamen():
3     def __init__(self):
4         # EN: Counter used for recursion control / ES: Contador usado para el control de la recursividad
5         self.con = 0
6
7     #Validate lowercase letters / Validar letras minúsculas
8     def validarMi(self, num):
9         # EN: Base case - if index reaches the end of string, it's valid / ES: Caso base - si el indice llega al final, es válido
10        if self.con >= len(num):
11            self.con = 0
12            return True
13        # EN: Check if the current character is lowercase (ASCII 97-122) / ES: Verifica si el carácter actual es minúscula (ASCII 97-122)
14        if ord(num[self.con])>=97 and ord(num[self.con])<=122:
15            self.con +=1
16            return self.validarMi(num) # EN: Recursive call / ES: Llamada recursiva
17        else:
18            self.con = 0
19            return False # EN: Return false if invalid character / ES: Retorna falso si hay un carácter inválido
20
21     #Validate uppercase letters / Validar letras mayúsculas
22     def validarMa(self, num):
23         if self.con >= len(num):
24             self.con = 0
25             return True
26         # EN: Check if current character is uppercase (ASCII 65-90) / ES: Verifica si el carácter actual es mayúscula (ASCII 65-90)
27         if ord(num[self.con])>=65 and ord(num[self.con])<=90:
28             self.con +=1
29             return self.validarMa(num)
30         else:
31             self.con = 0
32             return False
33
34     #Validate numbers / Validar números
35     def validarnu(self, num):
36         if self.con >= len(num):
37             self.con = 0
38             return True
39         # EN: Check if current character is a number (ASCII 48-57) / ES: Verifica si el carácter actual es un número (ASCII 48-57)
40         if ord(num[self.con])>=48 and ord(num[self.con])<=57:
41             self.con +=1
42             return self.validarnu(num)
43         else:
44             self.con = 0
45             return False
46
47

```

Ilustración 53 Código validaciones preexamen



Parcial 3

Programa 1 Pilas y colas

Este programa enseña el uso de Radiobutton para poder seleccionar la eliminación con los métodos de pilas o colas, además de poder ordenar los elementos de la lista de mayor a menos o viceversa

```

1  from tkinter import *
2  from tkinter import messagebox
3  from validaciones import Validar
4  import numpy as np
5
6  class Principal():
7      def __init__(self):
8          self.valid = Validar() # Instancia del validador / Validator instance
9          self.ven = Tk() # Crear ventana principal / Create main window
10         self.lista[] # Lista auxiliar / Auxiliary list
11         ancho = 300 # Ancho de la ventana / Window width
12         alto = 200 # Alto de la ventana / Window height
13         pantalla_ancho = self.ven.winfo_screenwidth() # Ancho de pantalla / Screen width
14         pantalla_alto = self.ven.winfo_screenheight() # Alto de pantalla / Screen height
15         x = (pantalla_alto/2)-(ancho/2) # Calcular posición X / Calculate X position
16         y = (pantalla_ancho/2)-(alto/2) # Calcular posición Y / Calculate Y position
17         self.ven.geometry(f"(ancho)x(alto)+(x{350}+y{350})") # Definir tamaño y posición / Set window size and position
18
19     def validarajx(self):
20         valor = self.datos.get() # Obtener el valor del campo de entrada / Get entry value
21         if (self.valid.ValidarNumeros(valor)): # Verifica si es un número / Check if it's a number
22             if (self.valid.ValidarIntradas(valor)): # Verifica si tiene dos dígitos / Check if it has two digits
23                 self.lista.insert(self.lista.size() + 1, valor) # Insertar valor en la lista / Insert value into list
24                 self.datos.delete(0,END) # Limpia caja de texto / Clear input box
25                 self.label.config(text="Elementos en la lista: "+str(self.lista.size())) # Actualizar contador / Update counter
26             else:
27                 messagebox.showerror("Incorrecto", "Solo se permite dos dígitos") # Error si supera 2 dígitos / Error if over 2 digits
28                 self.datos.delete(0,END)
29         else:
30             messagebox.showerror("Incorrecto", "No es un numero") # Error si no es número / Error if not a number
31             self.datos.delete(0,END)
32
33     def inicio(self):
34         self.datos = Entry(self.ven) # Campo de texto / Text entry field
35         self.datos.place(x=50,y=20)
36         self.listbox = Listbox(self.ven,selectmode="multiple") # Lista visual / Display list
37         self.listbox.place(x=100,y=50)
38         self.orden = StringVar(value="Pilas") # Orden inicial: stack
39         self.orden.set("Pilas") # Orden inicial: stack / Default order: stack
40         self.orden.trace("w",lambda *args: self.mostrar()) # Cambiar orden / Change order
41         Radiobutton(self.ven,text="Pilas",variable=self.orden,value="Pilas").place(x=50,y=50) # Radio para pilas / Stack option
42         Radiobutton(self.ven,text="Colas",variable=self.orden,value="Colas").place(x=100,y=50) # Radio para colas / Queue option
43         Radiobutton(self.ven,text="Mayor",variable=self.orden,value="Mayor").place(x=50,y=135) # Orden descendente / Descending order
44         Radiobutton(self.ven,text="Menor",variable=self.orden,value="Menor").place(x=100,y=135) # Orden ascendente / Ascending order
45         Button(self.ven,text="Limpiar",command=lambda:self.limpiar()).place(x=90,y=80,width=55) # Botón para validar / Validate button
46         Button(self.ven,text="Eliminar",command=lambda:self.eliminar()).place(x=90,y=150,width=55) # Botón eliminar / Delete button
47         Button(self.ven,text="Ordenar",command=lambda:self.ordenar()).place(x=90,y=150,width=55) # Botón ordenar / Sort button
48         self.lista = listbox(self.ven,height=10, width=15, bg="grey",
49                             activestyle="dotbox", fg="Black") # Lista visual / Display list
50         self.lista.place(x=100,y=20)
51         self.label = Label(self.ven, text="Elementos en la lista: 0") # Etiqueta contador / Counter label
52         self.label.place(x=50,y=100)
53         self.ven.mainloop() # Iniciar bucle principal / Start main loop
54
55     def limpiar(self):
56         if self.lista.size()<0:
57             messagebox.showerror("Error", "La lista esta vacía") # Mensaje si está vacía / Error if list is empty
58             return
59         if self.modo.get() == "Pilas":
60             # Último que entra, primero que sale / last in, first out (stack)
61             self.lista.delete(self.lista.size()-1)
62         else:
63             # Primero que entra, primero que sale / First in, first out (queue)
64             self.lista.delete(0)
65         self.label.config(text="Elementos en la lista: "+str(self.lista.size())) # Actualizar etiqueta / Update label
66
67     def eliminar(self):
68         self.lis = list(self.lista.get(0,END)) # Obtener lista actual / Get current list
69         if len(self.lis)<0:
70             messagebox.showerror("Error", "La lista esta vacía") # Error si lista vacía / Error if list empty
71         if self.orden.get() == "Mayor":
72             # Método de selección (orden descendente) / selection sort (descending)
73             p = 0
74             for i in range(0,len(self.lis)):
75                 aux = int(self.lis[i])
76                 pi =
77                 for x in range(i, len(self.lis)):
78                     if aux < int(self.lis[x]): # Buscar el mayor / Find the largest
79                         aux = int(self.lis[x])
80                         p =
81                         self.lis[p] = self.lis[i]
82                         self.lis[i] = str(aux)
83                         self.lista.delete(0,END)
84                         for i in self.lis:
85                             self.lista.insert(self.lista.size()+1,i) # Insertar ordenada / Insert sorted list
86             else:
87                 # Método burbuja (orden ascendente) / bubble sort (ascending)
88                 for i in range(0, len(self.lis)):
89                     for x in range(0, len(self.lis)-i):
90                         if self.lis[x]>self.lis[x+1]: # Comparar e intercambiar / Compare and swap
91                             self.lis[x], self.lis[x+1] = self.lis[x+1], self.lis[x]
92                             self.lista.delete(0,END)
93                             for i in self.lis:
94                                 self.lista.insert(self.lista.size()+1,i) # Insertar ordenada / Insert sorted list
95
96     if __name__=="__main__":
97         app = Principal() # Crear instancia / Create instance
98         app.inicio() # Ejecutar aplicación / Run app
99

```

Ilustración 54 Código programa 1



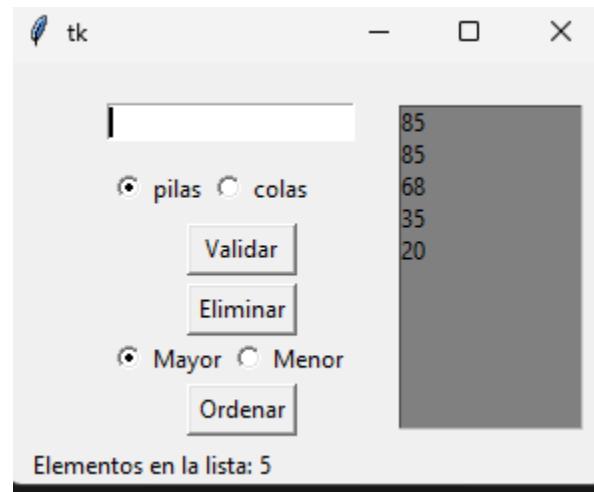


Ilustración 55 Ejecución programa 1



Programa 2 RFC

Este programa genera el RFC con los datos ingresado en las cajas de texto, una vez validado los datos lo agrega a una lista con la posibilidad de eliminar registros con métodos de colas o pilas

```

1  from tkinter import *
2  from tkinter import messagebox
3  from validaciones import Validar
4
5  class RFC:
6      def __init__(self):
7          self.val = Validar() # Instancia de la clase validaciones / Validator instance
8          self.ventana = Tk() # Ventana principal / Main window
9          self.lista[] # Lista auxiliar (no usada directamente) / Auxiliary list
10         ancho = 550 # Ancho de ventana / Window width
11         alto = 250 # Alto de ventana / Window height
12         pantalla_ancho = self.ventana.winfo_screenwidth() # Ancho de pantalla / Screen width
13         pantalla_alto = self.ventana.winfo_screenheight() # Alto de pantalla / Screen height
14         x = (pantalla_ancho//2)- (ancho//2) # Posición X centrada / Center X position
15         y = (pantalla_ancho//2)- (alto//2) # Posición Y centrada / Center Y position
16         self.ventana.geometry(f'{ancho}x{alto}+{x+350}+{y-350}') # Posicionar ventana / Position window
17
18     def inicio(self):
19         # ----- Cajas de texto / Input boxes -----
20         self.nombre = Entry(self.ventana) # Caja para nombre / Name entry
21         self.nombre.place(x=20,y=60,width=100)
22         self.paterno = Entry(self.ventana) # Caja para apellido paterno / Paternal surname
23         self.paterno.place(x=140,y=60,width=100)
24         self.materno = Entry(self.ventana) # Caja para apellido materno / Maternal surname
25         self.materno.place(x=260,y=60,width=100)
26         self.dia = Entry(self.ventana) # Dia / Day
27         self.dia.place(x=20,y=130,width=100)
28         self.mes = Entry(self.ventana) # Mes / Month
29         self.mes.place(x=140,y=130,width=100)
30         self.ano = Entry(self.ventana) # Año / Year
31         self.ano.place(x=260,y=130,width=100)
32
33         # ----- Etiquetas / Labels -----
34         Label(self.ventana, text="DATOS PERSONALES").place(x=125,y=10)
35         Label(self.ventana, text="Nombre").place(x=20,y=35)
36         Label(self.ventana, text="Apellido paterno").place(x=140,y=35)
37         Label(self.ventana, text="Apellido materno").place(x=260,y=35)
38         Label(self.ventana, text="Dia").place(x=20,y=105)
39         Label(self.ventana, text="Mes").place(x=140,y=105)
40         Label(self.ventana, text="Año").place(x=260,y=105)
41
42         # ----- Listbox / Lista visual -----
43         self.lista = Listbox(self.ventana, height=10, width=15, bg="grey",
44                             activestyle="dotbox", fg="Black")
45         self.lista.place(x=380, y=20,width=150)
46
47         # ----- Botones / Buttons -----
48         Button(self.ventana, text="Agregar", command=self.agregar).place(x=40,y=170,width=55) # Botón para agregar RFC / Add RFC button
49         Button(self.ventana, text="Eliminar", command=self.Eliminar).place(x=270,y=170,width=55) # Botón para eliminar / Delete button
50         self.modo = StringVar(value="Pilas") # Variable para el modo / Mode variable
51         Radiobutton(self.ventana, text="Pilas", variable=self.modo,value="Pilas").place(x=140,y=170) # Opción pila / Stack option
52         Radiobutton(self.ventana, text="Colas", variable=self.modo,value="Colas").place(x=190,y=170) # Opción cola / Queue option
53         self.ventana.mainloop() # Iniciar bucle de ventana / Start mainloop
54
55     def agregar(self):
56         # Obtener los datos desde las cajas / Get data from entries
57         nombre = self.nombre.get()
58         paterno = self.paterno.get()
59         materno = self.materno.get()
60         ano = self.ano.get()
61         dia = self.dia.get()
62         mes = self.mes.get()
63
64         # ----- Validaciones / Validations -----
65         if ano=="" or mes=="" or dia=="" or nombre=="" or paterno=="" or materno=="":
66             messagebox.showerror("Error", "Falta un dato") # Campo vacío / Missing field
67             self.nombre.delete(0,END)
68             self.paterno.delete(0,END)
69             self.materno.delete(0,END)
70             self.ano.delete(0,END)
71             self.dia.delete(0,END)
72             self.mes.delete(0,END)

```

Ilustración 56 Código programa 2



```

1 else:
2     if len(dia)==1:
3         dia = ('0'+dia) # Agrega cero si dia tiene un digito / Add leading zero if day has one digit
4     if len(ano)!=4 or len(mes)!=2 or len(dia)!=2:
5         messagebox.showerror("Error", "Formato de fecha erroneo") # Formato incorrecto / Wrong date format
6         self.ano.delete(0,END)
7         self.dia.delete(0,END)
8         self.mes.delete(0,END)
9     else:
10        if len(dia)==1:
11            dia = ('0'+dia)
12        # Verifica que los campos de fecha sean numéricos / Check that date fields are numeric
13        if (self.val.ValidarNumeros(ano))==False or (self.val.ValidarNumeros(dia))==False or (self.val.ValidarNumeros(mes))==False:
14            messagebox.showerror("Error", "No son numeros") # No numeros / Not numbers
15            self.ano.delete(0,END)
16            self.dia.delete(0,END)
17            self.mes.delete(0,END)
18        # Verifica que los nombres sean letras / Check that names are letters
19        if (self.val.ValidarNombre(paterno))==False or (self.val.ValidarNombre(materno))==False or (self.val.ValidarNombre(nombre))==False:
20            messagebox.showerror("Error", "Nombre incorrecto") # Nombre incorrecto / Invalid name
21            self.nombre.delete(0,END)
22            self.paterno.delete(0,END)
23            self.materno.delete(0,END)
24
25    else:
26        if int(mes)>12 or int(dia)>31:
27            messagebox.showerror("Error", "Fecha erronea") # Dia o mes fuera de rango / Invalid day or month
28            self.dia.delete(0,END)
29            self.mes.delete(0,END)
30
31    else:
32        if '-' in paterno:
33            paterno = paterno.split("-")[-1] # Si hay doble apellido, toma el ultimo / If double surname, take last
34            print(paterno)
35            bandera = False # Bandera no utilizada / Unused flag
36        if bandera == True:
37            if not (paterno[1] in ("a,e,i,o,u")):
38                paterno= paterno[0]+paterno[2]
39                bandera = False
40            else:
41                banderas = True
42
43        # Construcción del RFC / RFC construction
44        rfc = (f'{paterno[0:2]}{materno[0:1]}{nombre[0]}{ano[2:]}{mes}{dia}')
45        rfc = rfc.upper() # Convertir a mayúsculas / Convert to uppercase
46        self.lista.insert(END,rfc) # Agregar RFC a la lista / Add RFC to list
47
48        # Limpiar las cajas / Clear all entry boxes
49        self.ano.delete(0,END)
50        self.dia.delete(0,END)
51        self.mes.delete(0,END)
52        self.nombre.delete(0,END)
53        self.paterno.delete(0,END)
54        self.materno.delete(0,END)
55
56 def Eliminar(self):
57     # Eliminar elementos de la lista según el modo / Delete items based on mode
58     if self.lista.size()==0:
59         messagebox.showerror("Error", "La lista esta vacia") # Lista vacia / List is empty
60         return
61     if self.modo.get()=='Pilas':
62         # ultimo que entra, primero que sale / last in, first out (stack)
63         self.lista.delete(self.lista.size()-1)
64     else:
65         # primero que entra, primero que sale / first in, first out (queue)
66         self.lista.delete(0)
67
68 # Punto de entrada principal / Main entry point
69 if __name__=="__main__":
70     app = RFC() # Crear Instancia / Create instance
71     app.inicio() # Ejecutar aplicación / Run application

```

Ilustración 57 Código programa 2

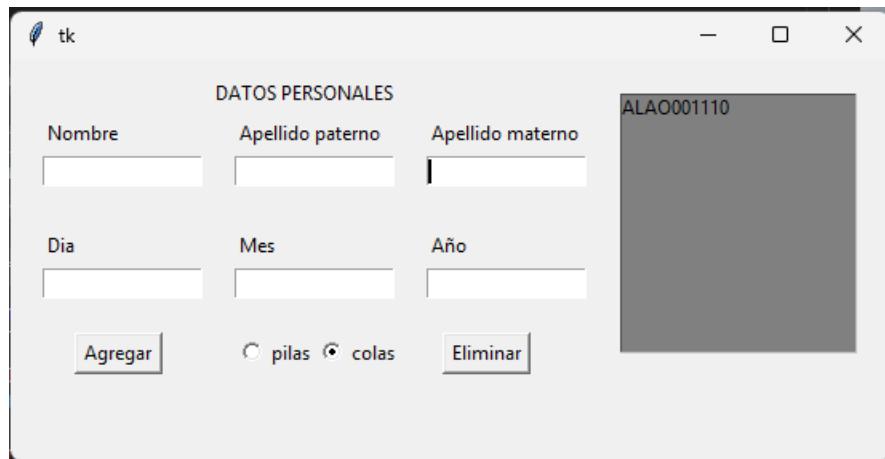


Ilustración 58 Ejecución programa 2



Programa 3: Registros con clave

Este programa ingresa información de usuarios para validarlos y crear una clave de registro para poder guardar los registros para mostrarlos en una lista

```

1  from tkinter import *
2  from tkinter import messagebox
3  from validaciones import Validar
4
5  class Principal:
6      def __init__(self):
7          self.val = Validar() # Instancia del validador / Validator instance
8          self.ventana = Tk() # Crear ventana principal / Create main window
9          self.listas = [] # Lista auxiliar (no usada directamente) / Auxiliary list
10         self.ventana.title('Práctica 3') # Titulo de la ventana / Window title
11
12         # Configuración de tamaño y posición / Window size and position
13         ancho = 350
14         alto = 250
15         pantalla_ancho = self.ventana.winfo_screenwidth() # Ancho de pantalla / Screen width
16         pantalla_alto = self.ventana.winfo_screenheight() # Alto de pantalla / Screen height
17         x = (pantalla_alto//2)- (ancho//2)
18         y = (pantalla_ancho//2)- (alto//2)
19         self.ventana.geometry(f'{ancho}x{alto}+{x+350}+{y-350}')
20
21         # ----- MÉTODOS PARA QUITAR PLACEHOLDER / REMOVE PLACEHOLDER METHODS -----
22         def quitar_placeholder1(self, event):
23             if self.nombre.get() == self.placeholder1: # Si el texto es el placeholder / If text equals placeholder
24                 self.nombre.delete(0, END)
25                 self.nombre.config(fg="black") # Cambiar color del texto / Change text color
26
27             def quitar_placeholder2(self, event):
28                 if self.telefono.get() == self.placeholder2:
29                     self.telefono.delete(0, END)
30                     self.telefono.config(fg="black")
31
32             def quitar_placeholder3(self, event):
33                 if self.domicilio.get() == self.placeholder3:
34                     self.domicilio.delete(0, END)
35                     self.domicilio.config(fg="black")
36
37             # ----- MÉTODOS PARA PONER PLACEHOLDER / ADD PLACEHOLDER METHODS -----
38             def poner_placeholder1(self, event):
39                 if self.nombre.get() == "": # Si está vacío / If empty
40                     self.nombre.insert(0, self.placeholder1)
41                     self.nombre.config(fg="gray")
42
43             def poner_placeholder2(self, event):
44                 if self.telefono.get() == "":
45                     self.telefono.insert(0, self.placeholder2)
46                     self.telefono.config(fg="gray")
47
48             def poner_placeholder3(self, event):
49                 if self.domicilio.get() == "":
50                     self.domicilio.insert(0, self.placeholder3)
51                     self.domicilio.config(fg="gray")
52
53             # ----- INICIO DE LA INTERFAZ / MAIN UI SETUP -----
54             def inicio(self):
55                 # ----- Caja de texto: nombre / Name input -----
56                 self.placeholder1 = "Nombre"
57                 self.nombre = Entry(self.ventana, fg="gray")
58                 self.nombre.insert(0, self.placeholder1)
59                 self.nombre.bind("<FocusIn>", self.quitar_placeholder1)
60                 self.nombre.bind("<FocusOut>", self.poner_placeholder1)
61                 # self.nombre.bind("<Return>", self.validarCaja) # Presionar Enter (opcional) / Optional Enter trigger
62                 self.nombre.place(x=10, y=10, width=100)
63
64                 # ----- Caja de texto: teléfono / Phone input -----
65                 self.placeholder2 = "Teléfono"
66                 self.telefono = Entry(self.ventana, fg="gray")
67                 self.telefono.insert(0, self.placeholder2)
68                 self.telefono.bind("<FocusIn>", self.quitar_placeholder2)
69                 self.telefono.bind("<FocusOut>", self.poner_placeholder2)
70                 # self.telefono.bind("<Return>", self.validarCaja)
71                 self.telefono.place(x=120, y=10, width=100)

```

Ilustración 59 Código programa 3



```

1  # ..... Caja de texto: domicilio / Address input .....
2  self.domicilio = Entry(self.ventana, "fg='gray")
3  self.domicilio.insert(0, self.placeholder3)
4  self.domicilio.bind("<focusin>", self.guitar_placeholder3)
5  self.domicilio.bind("<focusout>", self.poner_placeholder3)
6  self.domicilio.bind("<return>", self.validarCaja) # Enter ejecuta validacion / Enter triggers validation
7  self.domicilio.place(x=210, y=10, width=100)
8
9  # ..... Etiquetas y botones de selección de sexo / Gender selection .....
10 label(self.ventana,text="Sexo").place(x=10, y=10)
11 self.modo = StringVar(value="F") # valor por defecto femenino / default female
12 Radiobutton(self.ventana, text="M", variable=self.modo,value="M").place(x=10,y=50)
13 Radiobutton(self.ventana, text="F", variable=self.modo,value="F").place(x=10,y=70)
14
15 # ..... Lista para mostrar resultados / listbox to display results .....
16 self.lista = Listbox(self.ventana, height=8, width=50, bg="gray",
17                      activestyle="dotbox", fg="black")
18 self.lista.place(x=10, y=100)
19
20 # ..... Botón para agregar persona / Add person button .....
21 Button(self.ventana, text="Agregar", command=self.validar3).place(x=150,y=40,width=100,height=55)
22
23 self.ventana.mainloop() # Inicia la ventana / Start window
24
25 def agregar(self):
26     pass # Método reservado (no implementado) / Placeholder method (not implemented)
27
28 # ..... VALIDA Y AGREGAR DATOS / VALIDATE AND ADD DATA .....
29 def validarCaja(self,event=0):
30     # Verifica si los datos faltantes / Check for missing data
31     if self.nombre.get()==self.placeholder1 or (self.telefono.get()==self.placeholder2) or (self.domicilio.get()==self.placeholder3) or (self.domicilio.get()==""):
32         messagbox.showerror("Error", "Faltan datos")
33     else:
34         nombre = self.nombre.get()
35         telefono = self.telefono.get()
36         domicilio = self.domicilio.get()
37
38         # Validar formato del numero / Validate phone format
39         if len(telefono)!=10:
40             messagbox.showerror("Error", "Formato de numero incorrecto")
41             self.telefono.delete(0,END)
42
43         # Validar que sean solo numeros / Validate only numbers
44         if not (self.val.validarNumeros(telefono)):
45             messagbox.showerror("Error", "Solo se permiten numeros")
46             self.telefono.delete(0,END)
47
48         # Validar que el nombre tenga solo letras / Validate name contains only letters
49         if not self.val.ValidarNombre(nombre):
50             messagbox.showerror("Error", "Solo se permiten letras")
51             self.nombre.delete(0,END)
52         else:
53             # Asignar valor al sexo según selección / Assign gender
54             if self.modo.get()=="F":
55                 sexo = "Femenino"
56             else:
57                 sexo = "Masculino"
58
59             # Crear clave con iniciales y parte del domicilio / Create key with initials and address part
60             clave = nombre[0] + telefono[2:] + domicilio[2:]
61
62             # Concatenar todos los datos / Concatenate all data
63             persona = clave + " - " + nombre + " - " + telefono + " - " + domicilio + " - " + sexo
64
65             # Insertar en lista / Insert into listbox
66             self.lista.insert(END, persona)
67             persona =""
68
69             # Limpiar campos / Clear input fields
70             self.nombre.delete(0,END)
71             self.telefono.delete(0,END)
72             self.domicilio.delete(0,END)
73
74 # ..... EJECUCIÓN PRINCIPAL / MAIN EXECUTION .....
75 if __name__ == "__main__":
76     app = Principal() # Crear instancia / Create instance
77     app.inicio() # Ejecutar aplicación / Run application
78

```

Ilustración 60 Código programa 3

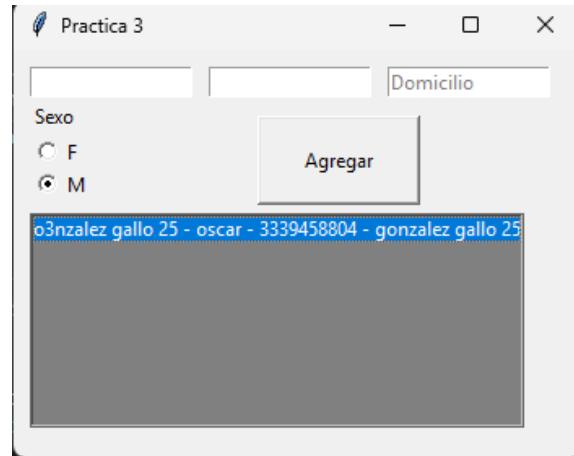


Ilustración 61 Ejecución programa 3



Programa 4: Treeview

Este programa utiliza la biblioteca treeview para crear una tabla con encabezados para poder agregar usuarios permitiendo agregar registros, eliminar o modificarlos

```

1  from tkinter import *
2  from tkinter import messagebox
3  from tkinter import ttk
4  import random
5  from validaciones import Validar
6
7  class Principal:
8      def __init__(self):
9          self.val = Validar() # instancia del validador
10         self.ventana = Tk()
11         self.ls=[]
12         self.ventana.title('Practica 4')
13
14         # Tamaño y posición de la ventana
15         ancho = 500
16         alto = 300
17         pantalla_ancho = self.ventana.winfo_screenwidth()
18         pantalla_alto = self.ventana.winfo_screenheight()
19         x = (pantalla_ancho // 2) - (ancho // 2)
20         y = (pantalla_alto // 2) - (alto // 2)
21         self.ventana.geometry(f'{ancho}x{alto}+{x}+{y}')
22
23         # Variables de control
24         self.con = 0 # Contador para generar claves
25         self.bandera = False # Bandera para modo edición
26         self.renglon = None # Guarda la fila seleccionada
27         self.index = "" # Guarda la clave del registro
28
29     def inicio(self):
30         # ----- Campos de texto -----
31         Label(self.ventana, text="Nombre").place(x=10, y=10)
32         self.nombre = Entry(self.ventana)
33         self.nombre.place(x=10, y=30, width=100)
34
35         Label(self.ventana, text="Edad").place(x=130, y=10)
36         self.edad = Entry(self.ventana)
37         self.edad.place(x=130, y=30, width=100)
38
39         Label(self.ventana, text="Correo").place(x=250, y=10)
40         self.correo = Entry(self.ventana)
41         self.correo.place(x=250, y=30, width=100)
42
43         # ----- Botones -----
44         Button(self.ventana, text="Aregar", command=self.AgregarElementos).place(x=380, y=20, width=100, height=30)
45         Button(self.ventana, text="Eliminar", command=self.eliminar).place(x=380, y=60, width=100, height=30)
46         Button(self.ventana, text="Seleccionar", command=self.validarCaja).place(x=380, y=100, width=100, height=30)
47
48         # ----- Tabla -----
49         columnas = ("Clave", "Nombre", "Correo", "Edad")
50         self.tabla = ttk.Treeview(self.ventana, columns=columnas, show="headings")
51         self.tabla.place(x=10, y=90, width=350, height=190)
52
53         # Encabezados y alineación
54         for col in columnas:
55             self.tabla.heading(col, text=col)
56             self.tabla.column(col, anchor="center", width=80)
57
58         # Scrollbars
59         scrollly = ttk.Scrollbar(self.ventana, orient="vertical", command=self.tabla.yview)
60         scrolllx = ttk.Scrollbar(self.ventana, orient="horizontal", command=self.tabla.xview)
61         self.tabla.configure(yscrollcommand=scrollly.set, xscrollcommand=scrolllx.set)
62         scrollly.place(x=360, y=90, height=190)
63         scrolllx.place(x=10, y=280, width=350)
64
65         self.ventana.mainloop()
66

```

Ilustración 62 Código programa 4



```

1
2     # ----- Seleccionar fila (modo edición) -----
3     def validarCaja(self):
4         seleccion = self.tabla.selection() # obtiene fila seleccionada
5         if not seleccion:
6             messagebox.showerror("Error", "Elige una fila")
7         else:
8             self.renglon = seleccion[0]
9             valores = self.tabla.item(self.renglon, "values")
10
11         # Guardamos los valores para edición
12         self.index = valores[0] # clave
13         self.nombre.delete(0, END)
14         self.edad.delete(0, END)
15         self.correo.delete(0, END)
16         self.nombre.insert(0, valores[1])
17         self.correo.insert(0, valores[2])
18         self.edad.insert(0, valores[3])
19         self.bandera = True # activamos modo edición
20
21     # ----- Agregar o actualizar elementos -----
22     def AgregarElementos(self):
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             return
26
27         nombre = self.nombre.get()
28         edad = self.edad.get()
29         correo = self.correo.get()
30
31         # Validaciones básicas
32         if not edad.isdigit():
33             messagebox.showerror("Error", "Edad debe ser numérica")
34             return
35         if not self.val.ValidarNombre(nombre):
36             messagebox.showerror("Error", "El nombre solo debe contener letras")
37             return
38
39         # Si no estamos editando, agregamos nuevo registro
40         if not self.bandera:
41             clave = str(self.con) + str(random.randint(1, 100)) + nombre[0:2].upper()
42             self.con += 1
43             self.tabla.insert("", "end", values=(clave, nombre, correo, edad))
44             messagebox.showinfo("Correcto", "Registro agregado correctamente")
45         else:
46             # Modo edición activado
47             print("Modo edición activado")
48             clave = self.index # mantener la misma clave
49             self.tabla.item(self.renglon, values=(clave, nombre, correo, edad))
50             self.bandera = False
51             self.renglon = None
52             messagebox.showinfo("Correcto", "Datos actualizados")
53
54         # Limpiar entradas después de agregar o editar
55         self.nombre.delete(0, END)
56         self.edad.delete(0, END)
57         self.correo.delete(0, END)
58
59     # ----- Eliminar elemento -----
60     def eliminar(self):
61         renglon = self.tabla.selection()
62         if not renglon:
63             messagebox.showerror("Error", "Elige una fila")
64         else:
65             self.tabla.delete(renglon)
66             messagebox.showinfo("Correcto", "Fila eliminada")
67
68 # ----- Ejecutar programa -----
69 if __name__ == "__main__":
70     app = Principal()
71     app.inicio()
72

```

Ilustración 63 Código programa 4

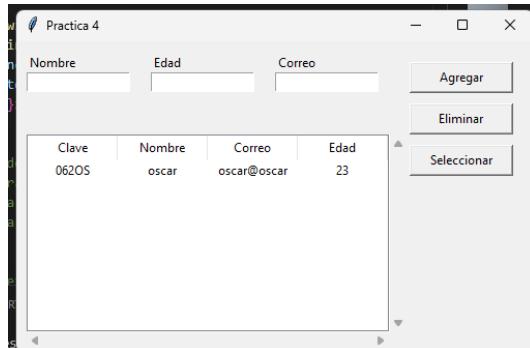


Ilustración 64 Ejecución programa 4



Validaciones

Validaciones utilizadas en los distintos programas

```
● ● ●

1  class Validar():
2      def __init__(self):
3          self.con = 0
4
5
6      def ValidarNumeros(self, num):
7          if self.con >= len(num):
8              self.con = 0
9              return True
10
11         if ord(num[self.con]) >= 47 and ord(num[self.con]) <= 58:
12             self.con += 1
13             return self.ValidarNumeros(num)
14         else:
15             self.con = 0
16             return False
17
18     def ValidarLetra(self, dato):
19         if ord(dato[0]) >= 65 and ord(dato[0])<= 90:
20             return True
21         else:
22             return False
23
24     def ValidarEntradas(self, dato):
25         if dato == "":
26             return False
27         if len(dato) == 2:
28             return True
29         else:
30             return False
31     def ValidarNombre(self, nom):
32         c = 0
33         for i in nom:
34             if (ord(i) >= 97 and ord(i) <= 122) or (ord(i) >= 97 and ord(i) <= 122) or (ord(i)==32) :
35                 c += 1
36         if c == len(nom):
37             return True
38         else:
39             return False
40
```

Ilustración 65 Código validaciones

