

# Manual de practicas



**Materia:** Sistema operativos

**Alumno:** Angel Gabriel Ramirez Aldrete

**Profesor:** Ricardo

**Carrera:** Informática 3



# Parcial 1

## Programa 1

```
1 |
2 a = int(input('Escribe un numero: ')) # Ask the user to enter a number | Solicita al usuario
3 print(a**2) # Calculate the power of the number (a²) | Calcula la potencia del número (a²)
4 print(a**(1/2)) # Calculate the square root of the number | Calcula la raíz cuadrada del número
5
6 # Operadores lógicos / Logical operators
7 '''
8 +, -, * # Addition, subtraction, multiplication | Suma, resta, multiplicación
9 / (exacta con decimales), //(sin decimales), ** (Potencia o elevar a), mod, and, or
10 / division with decimals, // integer division, ** exponentiation, mod modulus, and/or logical
11 / división exacta con decimales, // división entera, ** potencia, mod módulo, and/or operadores
12 '''
13
14 # Operadores diferenciales / Relational operators
15 '''
16 <, >, <=>, >=, <=, !=, ==, not
17 < less than, > greater than, <> not equal, >= greater or equal, <= less or equal, != not equal
18 < menor que, > mayor que, <> diferente, >= mayor o igual, <= menor o igual, != distinto, == igual
19 '''
20
21 PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1> & C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1\prog01.py"
Escribe un numero: 23
529
4.795831523312719
PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1> |
```

## Programa 2

```
1 |
2 a = [10] #arreglo
3 b = [] #lista
4 a[0] = 10
5 b = {'hola', 10, 10, 5, False, 'm', {1, 2, 3, 4}}
6
7 #ciclos y condiciones
8 if(len(a) > len(b)):
9     print('a es mayor')
10
11 else:
12     print('b es mayor')
13
14 for i in a:
15     print(a)
16
```

## Programa 3

```
1
2 # hacer un programa que lea 10 numeros y los almacene en un arreglo
3
4 # Se inicializa una lista 'a' con 9 elementos, todos con el valor 0.
5 # Es importante notar que la lista solo tiene 9 elementos, no 10.
6 a = [0, 0, 0, 0, 0, 0, 0, 0, 0]
7 # This initializes a list 'a' with 9 elements, all with the value 0.
8 # It's important to note that the list only has 9 elements, not 10.
9
10 # Este ciclo for intenta iterar 10 veces para pedir números al usuario.
11 # Sin embargo, debido a que la lista 'a' solo tiene 9 elementos (índices 0-8),
12 # el código generará un error de "índice fuera de rango" en la última
13 # iteración (cuando i = 9).
14 for i in range(10):
15     a[i] = int(input('Escribe un número \n'))
16 # This for loop attempts to iterate 10 times to ask the user for numbers.
17 # However, because list 'a' only has 9 elements (indexes 0-8),
18 # the code will produce an "index out of range" error on the last
19 # iteration (when i = 9).
20
21 # Este ciclo for recorre la lista 'a' y muestra cada uno de sus elementos.
22 # Si el código de arriba genera un error, este ciclo no se ejecutará.
23 for i in a:
24     print(i)
25 # This for loop iterates through the list 'a' and prints each of its elements.
26 # If the code above throws an error, this loop will not be executed.
```

PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1\progr03.py"

Escribe un número

87

Escribe un número

7

Escribe un número

4

Escribe un número

65

Escribe un número

76

Escribe un número

△ 0

## Programa 4

```
1
2 # Crea una lista para guardar los números válidos.
3 a = []
4 # Creates a list to store the valid numbers.
5
6 # Variables para la suma y el conteo de números.
7 s = 0
8 n = 0
9 # Variables for the sum and number count.
10
11 # Cadena con los dígitos para la validación de entrada.
12 numeros = "0123456789"
13 # String with digits for input validation.
14
15 # Bucle para obtener 10 números válidos.
16 while n < 10:
17     b = input('Escribe un número: ')
18 # Loop to get 10 valid numbers.
19
20     # Se inicializa un contador para los caracteres numéricos.
21     x = 0
22     for i in b:
23         if i in numeros:
24             x += 1
25     # A counter for numeric characters is initialized.
26
27     # Si la longitud de la entrada es igual al conteo de dígitos, es un número válido.
28     if len(b) == x:
29         a.append(int(b))
30         n += 1
31     else:
32         # If the input length equals the digit count, it's a valid number.
33
34         # Si no, se informa al usuario.
35         print('El valor no es un número válido')
36         # Otherwise, the user is informed.
37
```

```
37
38 # Recorre la lista, imprime los números y calcula la suma.
39 for i in a:
40     print(i)
41     s += i
42 # Loops through the list, prints the numbers, and calculates the sum.
43
44 # Muestra el resultado final.
45 print(f'La suma es {s}')
46 # Displays the final result.
47
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1> & C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1\prog04.py"
Escribe un número: 97
Escribe un número: 93
Escribe un número: 6
Escribe un número: 43
Escribe un número: 54
Escribe un número: 32
Escribe un número: 656
Escribe un número: 87
Escribe un número: 43
Escribe un número: 46
Escribe un número: 46
97
93
6
43
54
32
656
87
43
46
La suma es 1157
PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1> █
```

## Programa 5

```
1
2 # Hacer un programa que lea 10 datos.
3 # Si el dato es un número se almacenará en un arreglo (lista fija).
4 # Si es un carácter o varios, se pondrá en otra lista.
5 # Al final, se mostrará cuántos números y cuántos caracteres se ingresaron.
6
7 numeros = [0,0,0,0,0,0,0,0,0,0]
8 b = []
9 n = 0
10 e = 0
11
12
13 for i in range(10):
14     dato = input('Ingrese un dato:\n')
15
16
17     if dato.isdigit():
18         numeros[n] = int(dato)
19         n +=1
20
21     else:
22         b.append(dato)
23         e += 1
24
25 # Mostrar resultados
26 print("\nRESULTADOS")
27 print("Números en arreglo:")
28 for i in range(n):
29     print(numeros[i])
30
31 print("\nCaracteres de la lista:")
32 for i in b:
33     print(i)
34
35 print(f"\nCantidad de números: {n}")
36 print(f"Cantidad de caracteres: {e}")
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestres\Ingenieria informatica\Semestre 3\Estructura de datos>
Ingrese un dato:
dato+
Ingrese un dato:
esto
Ingrese un dato:
pepe
Ingrese un dato:
necio
Ingrese un dato:
menso
Ingrese un dato:
el
Ingrese un dato:
p+
Ingrese un dato:
s
Ingrese un dato:
ef
Ingrese un dato:
fd

RESULTADOS
Números en arreglo:

Caracteres de la lista:
dato+
esto
pepe
necio
menso
el
p+
s
ef
fd
+

```

```
p+
s
ef
fd

```

Cantidad de números: 0

Cantidad de caracteres: 10

PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1> |

## Programa 6

```
1
2  def resultados(numeros, b, n, e):
3      print('\nResultado')
4      print('Números en arreglo:')
5      for i in range(n):
6          print(numeros[i])
7
8      print('\nCaracteres de la lista:')
9      for i in b:
10         print(i)
11
12     print(f'\nCantidad de números: {n}')
13     print(f'Cantidad de caracteres: {e}')
14
15  def inicio():
16      # Global variables must be declared as such to be modified inside a function.
17      global numeros, b, n, e
18
19      print('hola mundo')
20      for i in range(10):
21          dato = input('Ingrese un dato:\n')
22
23          if dato.isdigit():
24              numeros[n] = int(dato)
25              n += 1
26          else:
27              b.append(dato)
28              e += 1
29
30      resultados(numeros, b, n, e)
31
32  # Initializing global variables outside of any function.
33  numeros = [0] * 10
34  b = []
35  n = 0
36  e = 0
37
```



```
38  ✓ if __name__ == "__main__":  
39      |     inicio()  
40
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estadística\Python> python3 1.py  
hola mundo  
Ingrese un dato:  
como esta  
Ingrese un dato:  
el  
Ingrese un dato:  
que  
Ingrese un dato:  
quiere  
Ingrese un dato:  
su  
Ingrese un dato:  
pizza  
Ingrese un dato:  
ya  
Ingrese un dato:  
no  
Ingrese un dato:  
es  
Ingrese un dato:  
pobre  
  
Resultado  
Números en arreglo:  
  
Caracteres de la lista:  
como esta  
el  
que  
quiere  
su  
pizza  
ya  
no  
es
```

```
ya  
no  
es  
pobre
```

```
Cantidad de números: 0  
Cantidad de caracteres: 10
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Estadística\Python>
```

## Programa 7

```
1
2 #hacer un programa que lea nombre edad y sexo de 5 personas,estos elementos deben estar
3 #dentro de una lista
4
5
6 def inicio():
7     l=0
8     while True:
9         aux = 0
10        b = input('Ingrese la edad:\n')
11        c = input('ingrese el sexo')
12        d = input('escribe el genero')
13        aux + 'nombre:' + b +'edad:' + c + 'genero' + d
14        list.append(aux)
15        c+=1
16        if c>=5:
17            break
18
19 print(list)
20
21
22 list =[]
23
24 if __name__ == "__main__": #metodo principal
25     inicio()
26
```

```
PS C:\Users\angel\Documents\Ingenieria informat
ments/Ingenieria informatica/Semestre 3/Estruct
Ingrese la edad:
20
ingrese el sexomasculino
escribe el genero m
```

## Programa 8

```
1 |
2 | #primera letramayuscukla
3 | #no se aceptan espacios
4 | n = 0
5 | c = [] # lista donde se guardarán las cadenas válidas
6 |
7 | while n < 5:
8 |     dato = input('Escribe una cadena:\n')
9 |
10 |    # Verificar si contiene espacios
11 |    if " " in dato:
12 |        print('No se aceptan espacios, intenta de nuevo')
13 |        continue
14 |
15 |    # Otra forma: usando replace
16 |    if len(dato) > 0:
17 |        primera = dato[0].upper()
18 |        dato = dato.replace(dato[0], primera, 1) # solo la primera ocurrencia
19 |
20 |    c.append(dato) # guardamos la cadena válida
21 |    n += 1
22 |
23 | print("\nLas cadenas ingresadas son:")
24 | print(c)
25 |
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestres\Ingenieria informatica\Semestre 3/Estructura de da
Escribe una cadena:
salsa
Escribe una cadena:
pepe
Escribe una cadena:
pan
Escribe una cadena:
d
Escribe una cadena:
ffd

Las cadenas ingresadas son:
['Salsa', 'Pepe', 'Pan', 'D', 'Ffd']
PS C:\Users\angel\Documents\Ingenieria informatica\Semestres\Ingenieria informatica\Semestre 3/Estructura de da
```

## Programa 9

```
1
2 # hacer que lea una cadena y que muestre en pantalla cuantos numeros tienes y
3 # cuantas mayusculas, cuantas minusculas y cuantos espacios
4
5 def inicio():
6     mi = 0 # minusculas
7     may = 0 # mayusculas
8     c = 0 # numeros
9     e = 0 # espacios
10
11     numero = "0123456789"
12     cadena = input('Escribe una cadena\n')
13
14     for i in cadena:
15         if i in numero: # si es número
16             print('es número')
17             c += 1
18         elif ord(i) == 32: # si es espacio
19             e += 1
20         elif 97 <= ord(i) <= 122: # si es minúscula
21             mi += 1
22         elif 65 <= ord(i) <= 90: # si es mayúscula
23             may += 1
24
25     print(f'Los números son: {c}')
26     print(f'Los espacios son: {e}')
27     print(f'Las mayúsculas son: {may}')
28     print(f'Las minúsculas son: {mi}')
29
30
31 if __name__ == '__main__': # método principal
32     inicio()
```

```
PS C:\Users\angel\Documents\Ingenieria inform
ments/Ingenieria informatica/Semestre 3/Estr
Escribe una cadena
da
Los números son: 0
Los espacios son: 0
Las mayúsculas son: 0
Las minúsculas son: 2
PS C:\Users\angel\Documents\Ingenieria infor
```

## Repaso 1

```
1
2 #instrucciones de entrada y salida
3 #print() o print(f)
4 #print('hola mundo')
5 #print(f'hola mundo numero {10}')
6 #entrada de datos
7 #input('escribe un numero')#se introduce solo letras
8 #casting para convertir a valores especificos
9 #f =0.0
10 #f = float(input('escribe numeros decimales'))
11 #a =0
12 #a = int(input('escribe un numero'))
13 #c =120
14 ##print(str(c))
15 #v =""
16 #v = str(c)
17 #nota solo las variables que no se introduce por teclado se obliga a inicializarlas.
18 #hacer un programa que lea el nombre precio de un producto el programa calculara
19 #el costo y el precio de venta
20 #costo involucra el 12% y el iva 16%
21 #while(true)
22 for i in range(0,5): #el rango valor inicial hasta el valor final sin incluirlo
23     precio = 12.55
24     nombre = input('Ingrese el nombre del producto:\n')
25     precio = float(input('ingrese el precio del producto: '))
26     costo = precio * 1.12
27     precioventa = costo * 1.16
28     print(f'el costo es {costo :.2f} y el precio de venta {precioventa:.2f}')
29     print(f'el costo es {costo} y el precio de venta {precioventa}')
30     #res = input('deseas otro numero (s/n)\n')
31     #if res == 'n' or res == 'N':
32     #    break
33
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Es
ments\Ingenieria informatica\Semestre 3\Estructura de datos\Prog
Ingrese el nombre del producto:
mango
ingrese el precio del producto: 20
el costo es 22.40 y el precio de venta 25.98
el costo es 22.400000000000002 y el precio de venta 25.984
Ingrese el nombre del producto:
█
```

## Repaso 2

```
repaso2.py > ...
1
2 a = 0
3 b = 0
4 c = 0
5 m = 0
6 r = 0
7 ra = 0.0
8 d = 0.0
9 x1 = 0.0
10 x2 = 0.0
11
12 # Aquí deberías pedir los valores de a, b, c
13 a = float(input("Introduce el valor de a: "))
14 b = float(input("Introduce el valor de b: "))
15 c = float(input("Introduce el valor de c: "))
16
17 p = b ** 2
18 m = 4 * a * c
19 r = p - m
20
21 if r > 0:
22     print('sí se puede, hay dos soluciones reales')
23     ra = r ** (1/2)
24     d = 2 * a
25     x1 = (-b + ra) / d
26     x2 = (-b - ra) / d
27     print(f'el valor de x1 es {x1:.2f} y el valor de x2 es {x2:.2f}')
28 else:
29     print('no se puede, no hay soluciones reales')
30
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestres\Ingenieria informatica/Semestre 3/Estructura de
Introduce el valor de a: 23
Introduce el valor de b: 86
Introduce el valor de c: 46
sí se puede, hay dos soluciones reales
el valor de x1 es -0.65 y el valor de x2 es -3.09
PS C:\Users\angel\Documents\Ingenieria informatica\Sem
```

## Repaso 3

```
1
2 #def validar(a):
3     # c = 0
4     #d = 0.0
5     #try:
6         # c = int(a)
7         # print('Es un valor numerico sin decimales')
8     #except ValueError:
9         # print('No es un valor numerico con decimales')
10
11     #try:
12         # d = float(a)
13         # print('Es un valor numerico con decimales')
14     #except ValueError:
15         # print('No es un valor con decimales')
16
17 #def leer():
18     # ord que obtiene el ascii del caracter
19     # isalpha para caracteres
20     # isdigit para numeros
21     # try except ValueError
22     # a = input('Escribe un dato o valor')
23     # validar(a)
24
25 # Hacer un programa que lea un dato y que lo almacene en un lista respetando su tipo de dato
26 def validar(a):
27     nf = 0
28     ne = 0
29     try:
30         ne = int(a)
31         return ne
32     except ValueError:
33         print('No es un entero')
34
35 def leer():
36     a = input('Escribe un dato \n')
37     dato = validar(a)
```

```
38     lista.append(dato)
39
40     lista = []
41
42     if __name__ == '__main__':
43         while(True):
44             leer()
45             res = input('Desea otro s/n')
46             if res == 'n' or res == 'N':
47                 print(lista)
48                 break
```

# Tarea 1

```
tarea1.py > ...
1 |
2 | def vocales(cad):
3 |     ba = False
4 |     be = False
5 |     bi = False
6 |     bo = False
7 |     bu = False
8 |
9 |     if 'a' in cad or 'A' in cad:
10 |         ba = True
11 |     if 'e' in cad or 'E' in cad:
12 |         be = True
13 |     if 'i' in cad or 'I' in cad:
14 |         bi = True
15 |     if 'o' in cad or 'O' in cad:
16 |         bo = True
17 |     if 'u' in cad or 'U' in cad:
18 |         bu = True
19 |
20 |     if ba == True and be == True and bi == True and bo == True and bu == True:
21 |         print("La cadena contiene todas las vocales")
22 |     else:
23 |         print("La cadena NO contiene todas las vocales")
24 |
25 |
26 | def minusculas(c):
27 |     cm = 0
28 |     print(c)
29 |     for i in c[1:]:
30 |         if ord(i) >= 97 and ord(i) <= 122:
31 |             cm += 1
32 |     if cm == len(c) - 1:
33 |         print(f"La cadena son minúsculas menos la primera {cm}")
34 |         vocales(c)
35 |     else:
36 |         print("Error: la cadena no cumple")
37 |
38 |
39 | def leer():
40 |     ce = 0 # antes no estaba inicializada
41 |     nc = ""
42 |     c = input("Escribe una cadena: ")
43 |
44 |     for i in c:
45 |         if ord(i) != 32:
46 |             ce += 1
47 |
48 |     if ce == len(c):
49 |         if c.isalpha():
50 |             minusculas(c)
51 |         else:
52 |             for i in c:
53 |                 if ord(i) >= 48 and ord(i) <= 57:
54 |                     pass
55 |                 else:
56 |                     nc += i
57 |             print(nc)
58 |             minusculas(nc)
59 |             print("Error: la cadena no cumple")
60 |     else:
61 |         print("Error: la cadena no cumple (espacios detectados)")
```



```

62
63
64 lista = []
65 if __name__ == "__main__":
66     while True:
67         leer()
68         lista.append(1) # algo para que cuente
69         if len(lista) >= 5:
70             break
71

```

```

PS C:\Users\angel\Documents\Ingenieria informatica\Semestr
ments\Ingenieria informatica/Semestre 3/Estructura de dato
Escribe una cadena: a
a
La cadena son minúsculas menos la primera 0
La cadena NO contiene todas las vocales
Escribe una cadena: g
g
La cadena son minúsculas menos la primera 0
La cadena NO contiene todas las vocales
Escribe una cadena: ghf
ghf
La cadena son minúsculas menos la primera 2
La cadena NO contiene todas las vocales
Escribe una cadena: wdf
wdf
La cadena son minúsculas menos la primera 2
La cadena NO contiene todas las vocales
Escribe una cadena: uyj
uyj
La cadena son minúsculas menos la primera 2
La cadena NO contiene todas las vocales
PS C:\Users\angel\Documents\Ingenieria informatica\Semestr

```

# Parcial 2

## Programa 1p2

```
prog1p2.py > ...
1
2 def inicio(num):
3     # Escribe una calificacion (Grade input)
4     a = int(input('Escribe una calificacion '))
5     # Incrementa el contador (Increment counter)
6     num += 1
7     # Añade la nota a la lista (Add grade to list)
8     lista.append(a)
9     # Condicion de fin (End condition: 5 grades)
10    if (num >= 5):
11        print()
12    else:
13        # Llamada recursiva (Recursive call)
14        return inicio(num)
15
16 # Lista para guardar notas (List to store grades)
17 lista = []
18 # Declarar variable global (Declare global variable)
19 global num
20 # Inicializa el contador (Initialize counter)
21 num = 0
22 # Ejecutar si el script es principal (Run if script is main)
23 if __name__ == '__main__':
24     # Iniciar la recoleccion de notas (Start grade collection)
25     inicio(num)
```

```
PS C:\Users\angel\Documents\Ingeneria info
Users/angel/Documents/Ingeneria info
Escribe una calificacion 9
Escribe una calificacion 6
Escribe una calificacion 8
Escribe una calificacion 9
Escribe una calificacion 10
```

## Programa 2p2

```
1
2 from validar import validacion # Importar la clase de validación (Import validation class)
3 val = validacion() # Crear instancia de la clase (Create class instance)
4
5 class Principal(): # Define la clase principal (Define the main class)
6     def __init__(self): # Constructor de la clase (Class constructor)
7         self.lista = [] # Lista para guardar notas (List to store grades)
8         self.num = 0 # Contador de notas (Grade counter)
9         self.a = "" # Variable para la entrada (Input variable)
10
11     def inicio(self):
12         # Solicita la calificación (Request the grade)
13         self.a = input("Escribe una calificación \n")
14
15         # Incrementa el contador (Increment counter)
16         if val.ValidarNumeros(self.a):
17             # Incrementa el contador (Increment counter)
18             self.num += 1
19             # Añade la nota a la lista (Add grade to list)
20             self.lista.append(int(self.a))
21
22             # Condición de fin (End condition: 5 grades)
23             if self.num >= 5:
24                 print(self.lista) # Imprime la lista (Print the list)
25                 # Calcula y muestra el promedio (Calculate and show average)
26                 print(f'El promedio es: {val.Promedio(self.lista)}')
27             else:
28                 # Llamada recursiva (Recursive call)
29                 self.inicio()
30         else:
31             # Mensaje de error (Error message)
32             print("No es un numero")
33             # Llama de nuevo para reintentar (Call again to retry)
34             self.inicio()
35
36 # Ejecutar si el script es principal (Run if script is main)
37 if __name__ == '__main__':
38     app = Principal() # Crear instancia de Principal (Create Principal instance)
39     app.inicio() # Iniciar el proceso (Start the process)
```

## Validaciones

```
1
2 class validacion(): # Define la clase de validacion y calculo (Define validation and calcula
3     def __init__(self): # Constructor de la clase (Class constructor)
4         self.suma = 0 # Inicializa la suma de valores (Initialize sum of values)
5         self.promedio = 0.0 # Inicializa el valor del promedio (Initialize average value)
6
7     def ValidarNumeros(self, valor): # Metodo para validar si la entrada es un numero (Metho
8         if valor.isdigit(): # Verifica si todos los caracteres son digitos (Checks if all ch
9             return True # Retorna verdadero (Returns True)
10        else:
11            return False # Retorna falso (Returns False)
12
13    def Promedio(self, lista): # Metodo para calcular el promedio (Method to calculate the a
14        # Recorre la lista para sumar los elementos (Iterate list to sum elements)
15        for i in lista:
16            self.suma += i # Acumula la suma total (Accumulate the total sum)
17        # Calcula el promedio (Calculate the average)
18        self.Promedio = self.suma / len(lista)
19        # Retorna el promedio calculado (Return the calculated average)
20        return self.promedio
21
```

```
PS C:\Users\angel\Documents\Ingenieria informatica\Semestre 3\Es
ments/Ingenieria informatica/Semestre 3/Es
Escribe una calificación
9
Escribe una calificación
7
Escribe una calificación
8
Escribe una calificación
10
Escribe una calificación
9
[9, 7, 8, 10, 9]
El promedio es: 0.0
PS C:\Users\angel\Documents\Ingenieria inf
```

## Programa 3p2

```
1
2 from tkinter import * # Importa todos los modulos de Tkinter (Import all Tkinter modules)
3 from tkinter import messagebox # Importa el módulo para cuadros de mensaje (Import message box module)
4
5 def Ventana(): # Define la función principal de la ventana (Define the main window function)
6     def revisar(): # Define la función para revisar credenciales (Define function to check credentials)
7         try:
8             # Obtener el texto del campo 'Usuario' (Get text from 'User' entry field)
9             u = str(us.get())
10            # Obtener el texto del campo 'Password' (Get text from 'Password' entry field)
11            p = str(pas.get())
12
13            # Comprobar credenciales (Check credentials)
14            if u == 'admin' and p == '12345':
15                # Mensaje de éxito (Success message)
16                messagebox.showinfo('Validación','Usuario y contraseña correcto')
17            else:
18                # Mensaje de error (Error message)
19                messagebox.showerror('Error','Usuario y/o contraseña incorrecto')
20        except ValueError:
21            # Manejar error si no hay datos (Handle error if no data is entered)
22            messagebox.showerror('Error','Introduce datos')
23
24        # Crear la ventana principal (Create the main window)
25        ven = Tk()
26        # Asignar título a la ventana (Set window title)
27        ven.title('Programa 1 con ventana')
28        # Definir tamaño y posición inicial (Define initial size and position)
29        ven.geometry('400x200')
30
31        # Etiqueta 'Usuario' (Label 'Usuario')
32        label = Label(ven, text = 'Usuario').pack(pady=10)
33        # Campo de entrada para el usuario (Entry field for username)
34        us = Entry(ven)
35        us.pack(pady=3)
36
37        # Etiqueta 'Password' (Label 'Password')
38        label = Label(ven, text = 'Password').pack(pady=10)
39        # Campo de entrada para la contraseña (Entry field for password)
40        pas = Entry(ven)
41        pas.pack(pady=3)
42
43        # Boton 'Aceptar' que llama a 'revisar' (Button 'Accept' that calls 'revisar')
44        boton = Button(ven, text='Aceptar', command=revisar).pack(pady=3)
45        # Inicia el ciclo de eventos de la ventana (Starts the window's event loop)
46        ven.mainloop()
47
48 # Ejecutar si el script es principal (Run if script is main)
49 if __name__ == '__main__':
50     Ventana() # Llamar a la función de la ventana (Call the window function)
```

Programa 1 con ventana

Usuario

angel

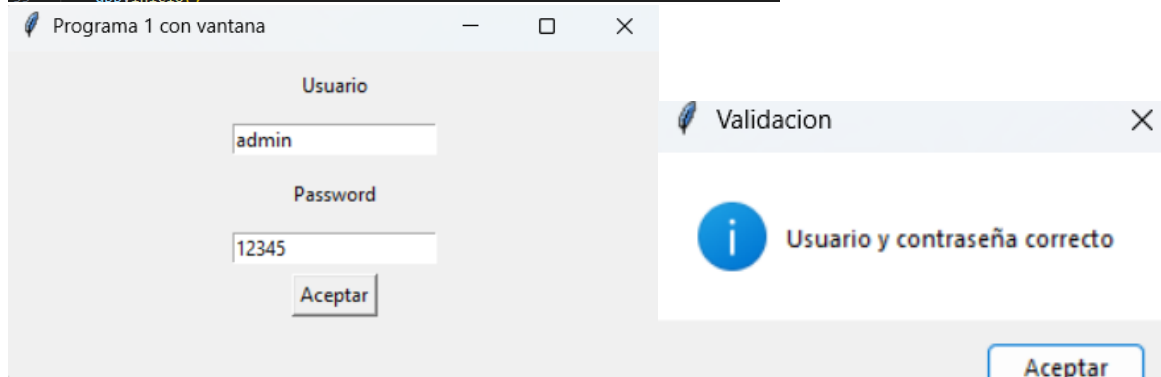
Password

23456

Aceptar

## Programa 4p2

```
1 from tkinter import * # Importa todos los módulos de Tkinter (Import all Tkinter modules)
2 from tkinter import messagebox # Importa el módulo para cuadros de mensaje (Import message box module)
3
4 class Ventana(): # Define la clase de la ventana (Define the window class)
5     def __init__(self): # Constructor de la clase (class constructor)
6         self.ven = Tk() # Crear la ventana principal como atributo (Create main window as attribute)
7         # Asignar título a la ventana (Set window title)
8         self.ven.title('Programa 1 con ventana')
9         # Definir tamaño inicial (Define initial size)
10        self.ven.geometry('400x200')
11
12    def inicio(self):
13        # Etiqueta 'Usuario' (Label 'Usuario')
14        label = Label(self.ven, text = 'Usuario').pack(pady=10)
15        # Campo de entrada (Entry field)
16        self.us = Entry(self.ven)
17        self.us.pack(pady=3)
18
19        # Etiqueta 'Password' (Label 'Password')
20        label = Label(self.ven, text = 'Password').pack(pady=10)
21        # Campo de entrada (Entry field)
22        self.pas = Entry(self.ven)
23        self.pas.pack(pady=3)
24
25        # Boton 'Aceptar' que llama a 'self.revisar' (Button 'Accept' calls 'self.revisar')
26        # El comando llama al metodo de la clase (Command calls class method)
27        boton = Button(self.ven, text='Aceptar', command=self.revisar).pack(pady=3)
28
29        # Inicia el ciclo de eventos (Starts the event loop)
30        self.ven.mainloop()
31
32    def revisar(self): # Metodo para revisar credenciales (Method to check credentials)
33        try:
34            # Obtener usuario del atributo self.us (Get user from self.us attribute)
35            u = str(self.us.get())
36            # Obtener password del atributo self.pas (Get password from self.pas attribute)
37            p = str(self.pas.get())
38
39            # Comprobar credenciales (Check credentials)
40            if u == 'admin' and p == '12345':
41                # Mensaje de exito (Success message)
42                messagebox.showinfo('Validacion','Usuario y contraseña correcto')
43            else:
44                # Mensaje de error (Error message)
45                messagebox.showerror('Error','Usuario y/o contraseña incorrecto')
46        except ValueError:
47            # Manejar error si no hay datos (Handle error if no data is entered)
48            messagebox.showerror('Error','Introduce datos')
49
50    # Ejecutar si el script es principal (Run if script is main)
51    if __name__ == '__main__':
52        # Crear una instancia de la clase (Create an instance of the class)
53        app = Ventana()
54        # Iniciar la interfaz (Start the interface)
55        app.inicio()
```



# Programa 5p2

```
1 from tkinter import *
2 from tkinter import messagebox
3
4 class Principal():
5     # Constructor de la clase (Class constructor)
6     def __init__(self):
7         self.ven = Tk() # Crear la ventana principal como atributo (Create main window as attribute)
8         # Asignar título a la ventana (Set window title)
9         self.ven.title('Programa 5 con ventana')
10        # Definir tamaño inicial (Define initial size)
11        self.ven.geometry('600x250')
12        self.lista = [] # Lista para almacenar los promedios calculados (List to store the calculated averages)
13        self.inicio() # Llamar al método para configurar la interfaz (Call the method to set up the interface)
14
15        # Método para sumar todos los elementos de la lista (Method to sum all elements in the list)
16        def sumar(self):
17            s = 0
18            for i in self.lista:
19                s += i
20            return s
21
22        # Método para calcular el promedio de 4 números y el promedio general (Method to calculate the average of 4 numbers and the
23        def promediar(self):
24            try:
25                # Obtener y convertir los valores de las cajas de texto a flotantes (Get and convert the values from the text boxes
26                a = float(self.n1.get())
27                b = float(self.n2.get())
28                c = float(self.n3.get())
29                d = float(self.n4.get())
30
31                # Calcular el promedio de los 4 números (Calculate the average of the 4 numbers)
32                pro = (a + b + c + d) / 4
33
34                # Actualizar la etiqueta l6 con el promedio calculado (Update label l6 with the calculated average)
35                self.l6.config(text=str(pro))
36
37                # Añadir el promedio a la lista (Add the average to the list)
38                self.lista.append(pro)
39
40                # Actualizar la etiqueta l7 con el contenido actual de la lista (Update label l7 with the current content of the list)
41                self.l7.config(text=str(self.lista))
42
43                # Limpiar las cajas de texto (Clear the text boxes)
44                self.n1.delete(0, END)
45                self.n2.delete(0, END)
46                self.n3.delete(0, END)
47                self.n4.delete(0, END)
48
49                # Calcular la suma de todos los promedios en la lista (Calculate the sum of all averages in the list)
50                suma = self.sumar()
51                print(suma) # Imprimir la suma en la consola (Print the sum to the console)
52
53                # Calcular el promedio general (Average of the averages) (Calculate the general average)
54                p = suma / len(self.lista)
55
56                # Actualizar la etiqueta l8 con el promedio general (Update label l8 with the general average)
57                self.l8.config(text=f'Promedio general: {str(p)}')
58
59            except ValueError:
60                # Capturar error si algún dato no es numérico (Catch error if any data is not numeric)
61                messagebox.showerror('Validacion', 'Algún dato no es número')
62                # Limpiar las cajas de texto en caso de error (Clear the text boxes in case of error)
63                self.n1.delete(0, END)
64                self.n2.delete(0, END)
65                self.n3.delete(0, END)
66                self.n4.delete(0, END)
67
68        # Método para cerrar la ventana (Method to close the window)
69        def salir(self):
70            self.ven.destroy()
71
72        # Método para configurar la interfaz de usuario (Method to set up the user interface)
73        def inicio(self):
74            # Etiquetas para pedir números (Labels to ask for numbers)
75            l1 = Label(self.ven, text="Escribe un número").place(y=10, x=20) # y=filas, x=columnas
76            l2 = Label(self.ven, text="Escribe un número").place(y=50, x=20)
77
78            # Cajas de texto (Entry widgets) para la entrada de números (for number input)
79            self.n1 = Entry(self.ven)
80            self.n1.place(y=10, x=130)
81            self.n2 = Entry(self.ven)
82            self.n2.place(y=50, x=130)
83
84            # Más etiquetas y cajas de texto (More labels and Entry widgets)
85            l3 = Label(self.ven, text="Escribe un número").place(y=90, x=20)
86            l4 = Label(self.ven, text="Escribe un número").place(y=130, x=20)
87            self.n3 = Entry(self.ven)
88            self.n3.place(y=90, x=130)
89            self.n4 = Entry(self.ven)
90            self.n4.place(y=130, x=130)
91
92            # Etiqueta fija para mostrar el promedio (Fixed label to show the average)
93            l5 = Label(self.ven, text="Promedio: ").place(y=150, x=130)
94
95            # Etiqueta que mostrará el resultado del promedio individual (Label that will show the result of the individual average)
96            self.l6 = Label(self.ven, text="0.0")
97            self.l6.place(y=150, x=200)
98
```

```
99 # Botones de la interfaz (Interface buttons)
100 b1 = Button(self.ven, text="Promedio", command=self.promediar).place(y=50, x=300) # Botón para calcular el promedio (Button to calculate the average)
101 b2 = Button(self.ven, text="Salir", command=self.salir).place(y=90, x=300) # Botón para salir (Button to exit)
102
103 # Etiqueta que mostrará la lista de promedios (Label that will show the list of averages)
104 self.l7 = Label(self.ven, text="[]")
105 self.l7.place(y=170, x=200)
106
107 # Etiqueta que mostrará el promedio general (Label that will show the general average)
108 self.l8 = Label(self.ven, text="Promedio general: 0.0")
109 self.l8.place(y=190, x=200)
110
111 # Iniciar el bucle principal de la aplicación (Start the application's main loop)
112 self.ven.mainloop()
113
114 # Bloque principal para ejecutar la aplicación (Main block to run the application)
115 if __name__ == '__main__':
116     APP = Principal()
```

Programa 5 con ventana

Escribe un número

Escribe un número  Promedio

Escribe un número  Salir

Escribe un número

Promedio: 49.5  
[315.5, 287.75, 346.5, 49.5]  
Promedio general: 249.8125

## Programa 6p2

```
1 from tkinter import *
2 from tkinter import messagebox
3 import random
4
5 class principal():
6     # Constructor de la clase
7     # Class constructor
8     def __init__(self):
9         # Crear la ventana principal de Tkinter
10        # Create the main Tkinter window
11        self.ven = Tk()
12        # Establecer el título de la ventana
13        # Set the window title
14        self.ven.title("Programa 6 con ventana GRID")
15        # Establecer el tamaño de la ventana
16        # Set the window size
17        self.ven.geometry('450x300') # Aumentado el alto un poco para mejor visibilidad / Increased height slightly for better visibility
18
19        # Variables de instancia para almacenar los números de entrada
20        # Instance variables to store the input numbers
21        self.a = 0
22        self.b = 0
23
24        # Lista para almacenar los números agregados
25        # List to store the added numbers
26        self.lista = []
27
28        # Variables auxiliares para encontrar el mayor y el menor (se usan min/max)
29        # Auxiliary variables for finding the maximum and minimum (min/max are used)
30        self.aux_mayor = 0 # Usar un nombre más descriptivo / Use a more descriptive name
31        self.aux_menor = 0 # Inicializado a 0, se actualizará en agregar() / Initialized to 0, will be updated in agregar()
32
33        # Inicializar contador para la función mayor (si se usa recursión)
34        # Initialize counter for the mayor function (if recursion is used)
35        self.cont = 0
36
```

```

37 # Función para configurar los elementos de la interfaz gráfica
38 # Function to set up the graphical interface elements
39 def inicio(self):
40     # Etiqueta de título
41     # Title label
42     l1 = Label(self.ven, text="Programa 9")
43     l1.grid(row=1, column=2, columnspan=2) # Uso del gestor grid (Use of grid manager)
44
45     # Etiqueta e campo de entrada para el primer número
46     # Label and entry field for the first number
47     l2 = Label(self.ven, text="Escribe un número (A):")
48     l2.grid(row=3, column=1, padx=15, pady=10, sticky=W)
49     Label(self.ven, text="").grid(row=2, column=2) # Espaciador / Spacer
50     self.n1 = Entry(self.ven)
51     self.n1.grid(row=3, column=2, columnspan=2)
52
53     # Etiqueta e campo de entrada para el segundo número
54     # Label and entry field for the second number
55     l3 = Label(self.ven, text="Escribe otro número (B):")
56     l3.grid(row=5, column=1, padx=15, pady=5, sticky=W)
57     Label(self.ven, text="").grid(row=4, column=2) # Espaciador / Spacer
58     self.n2 = Entry(self.ven)
59     self.n2.grid(row=5, column=2, columnspan=2)
60
61     # Botones de acción
62     # Action buttons
63     b1 = Button(self.ven, text="Agregar", command=self.agregar, width=8)
64     b1.grid(row=6, column=1, pady=10)
65     b2 = Button(self.ven, text="Mayor", command=self.mayor, width=8)
66     b2.grid(row=6, column=2)
67     b3 = Button(self.ven, text="Menor", command=self.menor, width=8)
68     b3.grid(row=6, column=3, padx=10)
69     b4 = Button(self.ven, text="Salir", command=self.salir, width=8)
70     b4.grid(row=6, column=4, padx=15)
71

```

```

72 # Etiqueta para mostrar los elementos de la lista (texto)
73 # Label to display the list elements (text)
74 Label(self.ven, text="Elementos en lista:").grid(row=7, column=1, pady=15, sticky=W)
75 self.listaElementos = Label(self.ven, text="")
76 self.listaElementos.grid(row=7, column=2, columnspan=2, pady=15, sticky=W)
77
78 # Listbox para mostrar los elementos de la lista de forma gráfica
79 # Listbox to display the list elements graphically
80 Label(self.ven, text="Números Agregados:").grid(row=2, column=4, sticky=S)
81 self.listview = Listbox(self.ven, height=10, width=15, bg='grey', activestyle="dotbox", fg="red")
82 self.listview.grid(row=3, column=4, rowspan=4, padx=15)
83
84 # Iniciar el bucle principal de la ventana
85 # Start the main window loop
86 self.ven.mainloop()
87
88 # Función para encontrar y mostrar el número mayor en la lista
89 # Function to find and display the largest number in the list
90 def mayor(self):
91     if not self.lista: # Verificar si la lista está vacía / Check if the list is empty
92         print("Lista vacía")
93         messagebox.showerror("Error", "La lista está vacía. ¡Agrega números primero!")
94         return # Salir de la función si la lista está vacía / Exit the function if the list is empty
95
96     # Usar la función max() de Python para encontrar el mayor de forma eficiente
97     # Use Python's max() function to find the largest efficiently
98     el_mayor = max(self.lista)
99
100     # Mostrar el resultado
101     # Display the result
102     print(f'El mayor es {el_mayor}')
103     messagebox.showinfo("El Mayor", f"El número mayor es: {el_mayor}")
104

```



```

105 # Función para encontrar y mostrar el número menor en la lista
106 # Function to find and display the smallest number in the list
107 def menor(self):
108     if not self.lista: # Verificar si la lista está vacía / Check if the list is empty
109         print("Lista vacía")
110         messagebox.showerror("Error", "La lista está vacía. ¡Agrega números primero!")
111         return # Salir de la función si la lista está vacía / Exit the function if the list is empty
112
113     # Usar la función min() de Python para encontrar el menor de forma eficiente
114     # Use Python's min() function to find the smallest efficiently
115     el_menor = min(self.lista)
116
117     # Mostrar el resultado
118     # Display the result
119     print(f'El menor es {el_menor}')
120     messagebox.showinfo('El Menor', f"El número menor es: {el_menor}")
121
122 # Función para agregar los números de las entradas a la lista
123 # Function to add the numbers from the entries to the list
124 def agregar(self):
125     try:
126         # Intentar obtener los valores de las entradas como enteros
127         # Try to get the values from the entries as integers
128         a_val = self.n1.get()
129         b_val = self.n2.get()
130
131         # Verificar si las entradas no están vacías
132         # Check if the entries are not empty
133         if not a_val and not b_val:
134             messagebox.showwarning("Advertencia", "Ambos campos están vacíos. Escribe al menos un número.")
135             return
136
137         if a_val:
138             self.a = int(a_val)
139             self.lista.append(self.a)
140
141             self.listview.insert(END, self.a) # Insertar en el Listbox / Insert into the Listbox
142             self.n1.delete(0, END) # Limpiar la entrada / Clear the entry
143
144         if b_val:
145             self.b = int(b_val)
146             self.lista.append(self.b)
147             self.listview.insert(END, self.b) # Insertar en el Listbox / Insert into the Listbox
148             self.n2.delete(0, END) # Limpiar la entrada / Clear the entry
149
150         # Actualizar la etiqueta que muestra los elementos de la lista
151         # Update the label that shows the list elements
152         self.listaElementos.config(text=f"{self.lista}")
153         print(f"Lista actual: {self.lista}")
154
155         # Inicializar o actualizar aux_menor con el primer elemento si es necesario
156         # Initialize or update aux_menor with the first element if necessary
157         if len(self.lista) == 1:
158             self.aux_menor = self.lista[0]
159             self.aux_mayor = self.lista[0]
160
161     except ValueError:
162         # Manejar el error si la entrada no es un número
163         # Handle the error if the input is not a number
164         messagebox.showerror("Error", "Al menos un dato no es un número válido. Por favor, ingresa solo números enteros.")
165
166 # Función para salir de la aplicación
167 # Function to exit the application
168 def salir(self):
169     self.ven.destroy()
170
171 # Ejecutar la aplicación
172 # Run the application
173 if __name__ == '__main__':
174     app = principal()
175     app.inicio()

```

Programa 6 con ventana GRID

Programa 9

Escribe un número (A):


Escribe otro número (B):

Números Agregados


- 32
- 435
- 878
- 767
- 766
- 43
- 887
- 677

Elementos en Lista: [32, 435, 878, 767, 766, 43, 887, 677]

El Mayor

 El número mayor es: 887

El Menor

 El número menor es: 32

# Programa7p2

```
1 from tkinter import *
2 from tkinter import messagebox
3
4 class principal:
5     """Clase principal para la aplicación Tkinter.
6     Main class for the Tkinter application."""
7
8     def __init__(self):
9         self.ven = Tk()
10        self.ven.title('Programa 9 con ventana GRID - Mejorado')
11        # La geometría se deja como referencia, pero 'grid' maneja el diseño.
12        # Geometry is left as a reference, but 'grid' manages the layout.
13        self.ven.geometry('550x300')
14
15        # Inicializa la lista principal para almacenar los números.
16        self.lista = []
17        # Variables auxiliares que se mantienen aunque se usen min/max.
18        # Auxiliary variables that are kept even though min/max are used.
19        self.a = 0
20        self.b = 0
21
22    def inicio(self):
23        """Configura la interfaz gráfica usando el gestor de geometría grid.
24        Sets up the graphical interface using the grid geometry manager."""
25
26        # Título de la aplicación.
27        Label(self.ven, text="Programa 9", font=('Arial', 12, 'bold')).grid(row=0, column=0, columnspan=4, pady=10)
28
29        # === Entradas (columna 0 y 1) ===
30
31        # Entrada 1
32        Label(self.ven, text="Escribe un número 1:").grid(row=1, column=0, sticky='w', padx=5, pady=5)
33        self.n1 = Entry(self.ven, width=15)
34        self.n1.grid(row=1, column=1, padx=5, pady=5, sticky='ew')
35
36        # Título de la aplicación.
37        Label(self.ven, text="Programa 9", font=('Arial', 12, 'bold')).grid(row=0, column=0, columnspan=4, pady=10)
38
39        # === Entradas (columna 0 y 1) ===
40
41        # Entrada 1
42        Label(self.ven, text="Escribe un número 1:").grid(row=1, column=0, sticky='w', padx=5, pady=5)
43        self.n1 = Entry(self.ven, width=15)
44        self.n1.grid(row=1, column=1, padx=5, pady=5, sticky='ew')
45
46        # Entrada 2
47        Label(self.ven, text="Escribe un número 2:").grid(row=2, column=0, sticky='w', padx=5, pady=5)
48        self.n2 = Entry(self.ven, width=15)
49        self.n2.grid(row=2, column=1, padx=5, pady=5, sticky='ew')
50
51        # === Botones (columna 2) ===
52        Button(self.ven, text="Agregar", command=self.agregar, width=10).grid(row=1, column=2, padx=10, pady=5)
53        Button(self.ven, text="Mayor", command=self.mayor, width=10).grid(row=2, column=2, padx=10, pady=5)
54        Button(self.ven, text="Menor", command=self.menor, width=10).grid(row=3, column=2, padx=10, pady=5)
55        Button(self.ven, text="Salir", command=self.salir, width=10).grid(row=4, column=2, padx=10, pady=5)
56
57        # === Listbox y Display de lista (columna 3) ===
58        Label(self.ven, text="Elementos Agregados:", font=('Arial', 10)).grid(row=0, column=3, sticky='w', padx=5, pady=5)
59
60        self.listview = Listbox(self.ven, height=10, bg='lightgrey', fg='red', activestyle='dotbox')
61        # El listbox se expande vertical y horizontalmente.
62        # The listbox expands vertically and horizontally.
63        self.listview.grid(row=1, column=3, rowspan=4, sticky='nsew', padx=10, pady=5)
64
65        # Etiqueta para mostrar la lista interna de Python (debajo de las entradas).
66        # Label to show the internal Python list (below the entries).
67        Label(self.ven, text="Lista Interna:", font=('Arial', 10, 'italic')).grid(row=3, column=0, sticky='w', padx=5, pady=5)
68        self.listaElementos = Label(self.ven, text=f'{self.lista}', relief=SUNKEN, anchor='w')
69
70        self.listaElementos.grid(row=4, column=0, columnspan=2, sticky='ew', padx=5)
71
72        # Configura las expansiones de columnas y filas.
73        # Configures column and row expansions.
74        self.ven.grid_columnconfigure(3, weight=1)
75        self.ven.grid_rowconfigure(4, weight=1)
76
77        self.ven.mainloop()
78
79    def mayor(self):
80        """Encuentra y muestra el número mayor de la lista usando max().
81        Finds and displays the maximum number in the list using max()."""
82        # Verifica si la lista está vacía.
83        # Verifies if the list is empty.
84        if not self.lista:
85            messagebox.showerror("Error", "La lista está vacía. Agregue números primero. / The list is empty. Add numbers first.")
86            return
87
88        # Uso eficiente de la función integrada max().
89        # Efficient use of the integrated max() function.
90        numero_mayor = max(self.lista)
91        messagebox.showinfo("El Mayor / The Maximum", f'El número mayor en la lista es: {numero_mayor}')
92
93    def menor(self):
94        """Encuentra y muestra el número menor de la lista usando min().
95        Finds and displays the minimum number in the list using min()."""
96        # Verifica si la lista está vacía.
97        # Verifies if the list is empty.
98        if not self.lista:
99            messagebox.showerror("Error", "La lista está vacía. Agregue números primero. / The list is empty. Add numbers first.")
100            return
101
102        # Uso eficiente de la función integrada min().
103        # Efficient use of the integrated min() function.
104        numero_menor = min(self.lista)
105        messagebox.showinfo("El Menor / The Minimum", f'El número menor en la lista es: {numero_menor}')
```

```

100
101     # Agrega ambos a la lista interna.
102     self.lista.append(self.a)
103     self.lista.append(self.b)
104
105     # Agrega ambos al Listbox para visualización.
106     self.listview.insert(END, self.a)
107     self.listview.insert(END, self.b)
108
109     # Limpia los campos de entrada.
110     self.n1.delete(0, END)
111     self.n2.delete(0, END)
112
113     # Actualiza la etiqueta de la lista.
114     self.listaElementos.config(text=f"{self.lista}")
115
116 except ValueError:
117     # Manejo de error si el input no es un entero o está vacío.
118     messagebox.showerror("Error", "Ambos valores deben ser números enteros válidos.")
119
120 def salir(self):
121     """Cierra la ventana de la aplicación."""
122     self.ven.destroy()
123
124 if __name__ == '__main__':
125     # Crea y ejecuta la aplicación.
126     app = principal()
127     app.inicio()

```

Programa 9 con ventana GRID - Mejorado

**Programa 9** **Elementos Agregados:**

Escribe un número 1:

Escribe un número 2:

*Lista Interna:*

54  
98  
665  
65  
12  
15  
97  
34  
45  
87  
856  
348

El Mayor / The Maximum



El número mayor en la lista es: 856

Aceptar

El Menor / The Minimum



El número menor en la lista es: 12

Aceptar

# Parcial 3

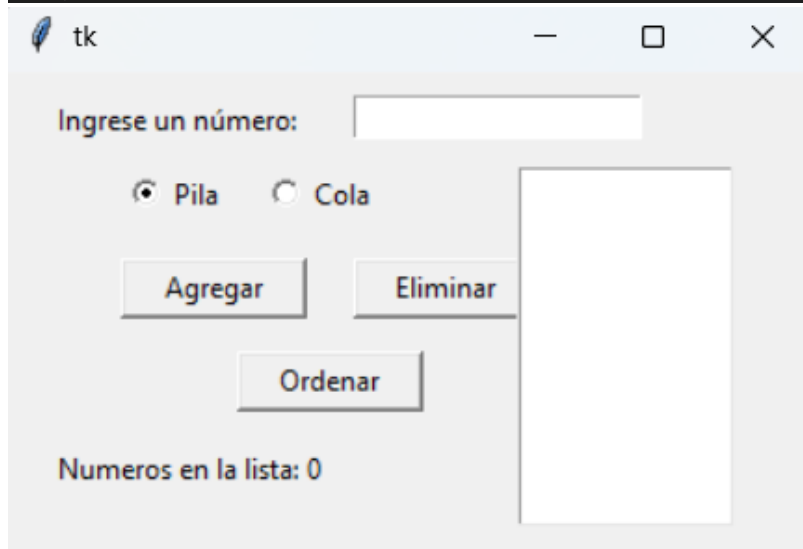
## Programa 1p3

```
1 from tkinter import *
2 from tkinter import messagebox
3 from validaciones import Validar
4 import numpy as np # ← corregido: era "es np"
5
6
7 class Principal:
8     def __init__(self):
9         self.val = Validar()
10        self.ven = Tk()
11        #self.ven.geometry("350x250")
12        ancho = 350
13        alto = 210
14        pantalla_ancho = self.ven.winfo_screenwidth()
15        pantalla_alto = self.ven.winfo_screenheight()
16        x = (pantalla_alto//2) - (ancho//2)
17        y = (pantalla_ancho//2) - (alto//2)
18        self.ven.geometry(f"{ancho}x{alto}+{x-1}+{y-400}")
19        self.lis = []
20        self.lista_datos = [] #lista lógica (Python)
21
22    def ValidarCaja(self):
23        valor = self.dato.get().strip()
24
25        # Validaciones
26        if self.val.validarNumeros(valor):
27            if self.val.validarEntrada(valor):
28                # Agregar a la lista lógica y al Listbox
29                self.lista_datos.append(valor) # ← corregido: antes decía self.lista_datos
30                self.lista.insert(END, valor)
31                self.dato.delete(0, END)
32                self.label.config(text=f'Numeros en la lista: {len(self.lista_datos)}')
33            else:
34                messagebox.showerror("Error", "Solo se permiten 2 dígitos")
35                self.dato.delete(0, END)
36        else:
37            messagebox.showerror("Error", "No es un número válido")
38
39        messagebox.showerror("Error", "No es un número válido")
40        self.dato.delete(0, END)
41
42    def eliminarDato(self):
43        if self.lista.size() <= 0:
44            messagebox.showwarning("Aviso", "La lista está vacía")
45            return
46
47        if self.modos.get() == "Pila":
48            # Último que entra, primero que sale (LIFO)
49            self.lista_datos.pop()
50            self.lista.delete(END)
51        else:
52            # Primero que entra, primero que sale (FIFO)
53            self.lista_datos.pop(0)
54            self.lista.delete(0)
55
56        self.label.config(text=f'Numeros en la lista: {len(self.lista_datos)}')
57
58    def ordenar(self):
59        self.lis = list(self.lista.get(0, END))
60        if len(self.lis) >= 0:
61            messagebox.showerror("Error", "Lista vacía")
62        else:
63            #burbuja
64            for i in range(0, len(self.lis)):
65                for x in range(0, len(self.lis)-1):
66                    if self.lis[x] > self.lis[x+1]:
67                        aux = self.lis[x]
68                        self.lis[x] = self.lis[x+1]
69                        self.lis[x+1] = aux
70            print(self.lis)
71            self.lista.delete(0, END)
72            for i in self.lis:
73                self.lista.insert(self.lista.size()+1, i)
```

```

72
73     #seleccion
74     '''if len(self.lista) <= 0:
75         self.lis = [int(i) for i in self.lista.get(0, END)]
76         #self.arreglo = np.array(self.lis)
77
78         for i in range(0, len(self.lis)):
79             aux = self.lis[i]
80             print(f'posible mayor {aux}')
81             for x in range(0, len(self.lis)):
82                 if aux < self.lis[x]:
83                     aux = self.lis[x]
84                     p = x
85             self.lis[p] = self.lis[i]
86             self.lis[i] = str(aux)
87         print(self.lis)
88         self.lista.delete(0,END)
89         for i in self.lis:
90             self.lista.insert(self.lista.size()+1, str(i))'''
91
92     def inicio(self):
93         Label(self.ven, text="Ingrese un número:").place(x=20, y=10)
94         self.dato = Entry(self.ven)
95         self.dato.place(x=150, y=10)
96
97         self.modos = StringVar(value="Pila")
98         Radiobutton(self.ven, text="Pila", variable=self.modos, value="Pila").place(x=50, y=40)
99         Radiobutton(self.ven, text="Cola", variable=self.modos, value="Colas").place(x=110, y=40)
100
101         Button(self.ven, text="Agregar", command=self.ValidarCaja, width=10).place(x=50, y=80)
102         Button(self.ven, text="Eliminar", command=self.eliminarDato, width=10).place(x=150, y=80)
103         Button(self.ven, text="Ordenar", command=self.ordenar, width=10).place(x=100, y=120)
104
105         self.label = Label(self.ven, text="Numeros en la lista: 0")
106
107         self.label.place(x=20, y=160)
108
109         self.lista = Listbox(self.ven, height=8, width=10, bg="white", fg="black", font=("Helvetica", 12))
110         self.lista.place(x=220, y=40)
111
112         self.ven.mainloop()
113
114 ✓ if __name__ == '__main__':
115     app = Principal()
116     app.inicio()

```



## Programa 2p3

## Programa 3p3

```
1
2  from tkinter import *
3  from tkinter import messagebox
4  from validar2p3 import validar2
5  import numpy as np
6
7  class Principal:
8      def __init__(self):
9          self.val = validar2()
10         self.ven = Tk()
11         self.ven.title('Pactica 3')
12         ancho = 350
13         alto = 250
14         pantalla_ancho = self.ven.winfo_screenwidth()
15         pantalla_alto = self.ven.winfo_screenheight()
16         x = (pantalla_alto//2)- (ancho//2)
17         y = (pantalla_ancho//2)- (alto//2)
18         self.ven.geometry(f"{ancho}x{alto}+{x-10}+{y-400}")
19
20
21     def quitar_placeholder1(self, event):
22         if self.nombre.get() == self.placeholder1:
23             self.nombre.delete(0, END)
24             self.nombre.config(fg="black")
25
26     def quitar_placeholder2(self, event):
27         if self.telefono.get() == self.placeholder2:
28             self.telefono.delete(0, END)
29             self.telefono.config(fg="black")
30
31     def quitar_placeholder3(self, event):
32         if self.domicilio.get() == self.placeholder3:
33             self.domicilio.delete(0, END)
34             self.domicilio.config(fg="black")
35
36     def poner_placeholder1(self, event):
37         if self.nombre.get() == "":
```

```

38         self.nombre.insert(0, self.placeholder1)
39         self.nombre.config(fg="gray")
40
41     def poner_placeholder2(self, event):
42         if self.telefono.get() == "":
43             self.telefono.insert(0, self.placeholder2)
44             self.telefono.config(fg="gray")
45
46     def poner_placeholder3(self, event):
47         if self.domicilio.get() == "":
48             self.domicilio.insert(0, self.placeholder3)
49             self.domicilio.config(fg="gray")
50
51     def inicio(self):
52         #caja de texto Nombre
53         self.placeholder = "Nombre"
54         self.nombre = Entry(self.ven, fg="gray")
55         self.nombre.insert(0, self.placeholder)
56         self.nombre.bind("<FocusIn>", self.quitar_placeholder1)
57         self.nombre.bind("<FocusOut>", self.poner_placeholder1)
58         #self.nombre.bind("<Return>", self.validarCaja)
59         self.nombre.place(x=50, y=10, width=100)
60         #caja de texto Telefono
61         self.placeholder = "Telefono"
62         self.telefono = Entry(self.ven, fg="gray")
63         self.telefono.insert(0, self.placeholder)
64         self.telefono.bind("<FocusIn>", self.quitar_placeholder2)
65         self.telefono.bind("<FocusOut>", self.poner_placeholder2)
66         #self.telefono.bind("<Return>", self.validarCaja)
67         self.telefono.place(x=120, y=30, width=100)
68         #caja de texto domicilio
69         self.placeholder = "Domicilio"
70         self.domicilio = Entry(self.ven, fg="gray")
71         self.domicilio.insert(0, self.placeholder)

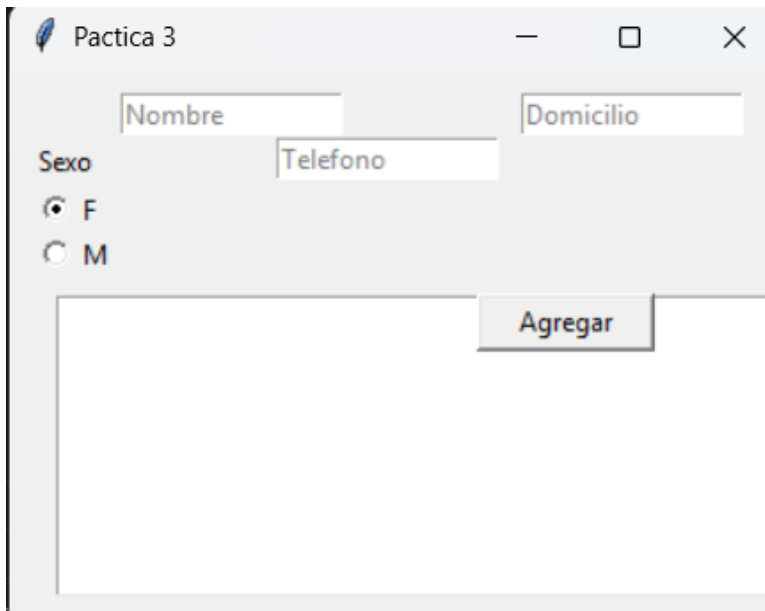
```

```

72         self.domicilio.bind("<FocusIn>", self.quitar_placeholder3)
73         self.domicilio.bind("<FocusOut>", self.poner_placeholder3)
74         self.domicilio.bind("<Return>", self.validarCaja)
75         self.domicilio.place(x=230, y=10, width=100)
76
77         Label(self.ven, text="Sexo").place(x=10, y=30)
78         self.mod0 = StringVar(value="F")
79         Radiobutton(self.ven, text="F", variable=self.mod0, value="F").place(x=10, y=50)
80         Radiobutton(self.ven, text="M", variable=self.mod0, value="M").place(x=10, y=70)
81         self.lista = Listbox(self.ven, height=7, width=40, bg="white", fg="black", font=("Helvetica", 12))
82         self.lista.place(x=20, y=100)
83         Button(self.ven, text="Agregar", command=self.validarCaja, width=10).place(x=210, y=100)
84         self.ven.mainloop()
85
86     def validarCaja(self, event):
87         if (self.nombre.get() == self.placeholder1 or self.telefono.get() == self.placeholder2 or
88             self.domicilio == self.placeholder3 or self.domicilio.get() == ""):
89             messagebox.showerror("Error", "Faltan datos")
90         else:
91             nombre = self.no (variable) telefono: Entry
92             telefono = self.telefono.get()
93             domicilio = self.domicilio.get()
94             if self.mod0.get() == "F":
95                 sexo = "Femenino"
96             else:
97                 sexo = "Masculino"
98             clave = nombre[0] + telefono[0] + domicilio[2:]
99             persona = nombre + " " + "+telefono+" " + "+domicilio+" " + "+sexo"
100             self.lista.insert(self.lista.size)
101
102 if __name__ == '__main__':
103     app = Principal()
104     app.inicio()

```





## Programa 4p3

```

1  from tkinter import *
2  from tkinter import messagebox
3  from tkinter import ttk
4  from validaciones import Validar
5  import numpy as np
6  import random
7
8  class Principal():
9      def __init__(self):
10         self.val = Validar()
11         self.ven = Tk()
12         self.ven.title('Practica 4')
13         #self.ven.geometry("500x300")
14         ancho = 500
15         alto = 300
16         ventana_alto = self.ven.winfo_screenwidth()
17         ventana_ancho = self.ven.winfo_screenheight()
18         x = (ventana_alto // 2) - (ancho // 2)
19         y = (ventana_ancho // 2) - (alto // 2)
20         self.ven.geometry(f"{ancho}x{alto}+{x}+{y-100}")
21         self.cont = 0
22         self.bandera = False
23         self.renglon = -1
24         self.index = 0
25
26     def validarCaja(self):
27         self.renglon = self.tabla.selection()
28         if not self.renglon:
29             messagebox.showerror("Error", "Elige una fila")
30         else:
31             valores = self.tabla.item(self.renglon, "values")
32             print(valores)
33             self.index = valores[0]
34             self.index = self.index[:len(self.index)-2]
35             print(self.index)
36             self.nombre.insert(0, valores[1])
37             self.edad.insert(0, valores[3])

```

```

38         self.correo.insert(0, valores[2])
39         self.bandera= True
40
41
42     def agregarElemento(self):
43         if (len(self.nombre.get())==0 or len(self.edad.get())== 0 or len(self.correo.get())== 0):
44             messagebox.showerror("Error", "Faltan datos")
45         else:
46             nombre = self.nombre.get()
47             edad = self.edad.get()
48             correo = self.correo.get()
49             if self.bandera == False:
50                 self.cont += 1
51                 clave = str(self.cont)+str(random.randint(1,100))+self.nombre.get()[0:2].upper()
52                 self.tabla.insert("", "end", values=(clave,nombre,correo,edad))
53                 self.nombre.delete(0,END)
54                 self.edad.delete(0,END)
55                 self.correo.delete(0,END)
56             else:
57                 clave = self.index+self.nombre.get()[0:2].upper()
58                 print("Modo edicion activado")
59                 self.tabla.item(self.renglon, values=(clave,nombre,correo,edad))
60                 self.nombre.delete(0,END)
61                 self.edad.delete(0,END)
62                 self.correo.delete(0,END)
63                 self.bandera = False
64                 self.renglon= -1
65                 messagebox.showinfo("Correcto", "Datos Actualizados")
66
67
68     def eliminar(self):
69         renglon = self.tabla.selection()
70         if not renglon:
71             messagebox.showerror("Error", "Elige una fila")
72         else:
73             self.tabla.delete(renglon)
74             messagebox.showinfo("Correcto", "Fila eliminada")
75
76
77     def inicio(self):
78         Label(self.ven, text="Nombre").place(x=10,y=10)
79         self.nombre = Entry(self.ven, fg="blue")
80         self.nombre.place(x=10, y=40, width=100)
81         Label(self.ven, text="Edad").place(x=130,y=10)
82         self.edad = Entry(self.ven, fg="green")
83         self.edad.place(x=125, y=40, width=100)
84         Label(self.ven, text="correo").place(x=250,y=10)
85         self.correo = Entry(self.ven, fg="purple")
86         self.correo.place(x=240, y=40, width=100)
87         Button(self.ven, text="Agregar", command=self.agregarElemento, width=10).place(x=380,y=50, width=100,height=30)
88         Button(self.ven, text="Eliminar", command=self.eliminar, width=10).place(x=380,y=90, width=100,height=30)
89         Button(self.ven, text="Seleccionar", command=self.validarCaja, width=10).place(x=380,y=130, width=100,height=30)
90         #datagrid
91         columnas = ("Clave", "Nombre", "Correo", "Edad")
92         self.tabla = ttk.Treeview(self.ven, columns= columnas, show="headings")
93         self.tabla.place(x=10, y=100, width=350,height=190)
94         for col in columnas:
95             self.tabla.heading(col, text=col)
96             self.tabla.column(col, anchor="center", width=30)
97         scrolly = ttk.Scrollbar(self.ven, orient="vertical", command=self.tabla.yview)
98         scrollx = ttk.Scrollbar(self.ven, orient="horizontal", command=self.tabla.xview)
99         scrolly.place(x=360,y=90,height=200)
100         scrollx.place(x=10,y=280, width=350)
101
102         self.ven.mainloop()
103
104 if __name__ == '__main__':
105     app = Principal()
106     app.inicio()

```

## Validaciones

```

1 class Validar:
2     def __init__(self):
3         self.con = 0
4
5     def validarNumeros(self, num):
6         """Verifica que la cadena contenga solo números."""
7         if num == "":
8             return False
9
10        if self.con >= len(num):
11            self.con = 0
12            return True
13
14        if 48 <= ord(num[self.con]) <= 57: # '0' a '9'
15            self.con += 1
16            return self.validarNumeros(num)
17        else:
18            self.con = 0
19            return False
20
21    def validarLetra(self, dato):
22        """Verifica que el primer carácter sea una letra mayúscula."""
23        if dato == "":
24            return False
25
26        if 65 <= ord(dato[0]) <= 90: # 'A' a 'Z'
27            return True
28        else:
29            return False
30
31    def validarEntrada(self, dato):
32        """Verifica que la cadena tenga máximo 2 dígitos."""
33        if dato == "":
34            return False
35        return len(dato) <= 2

```

## Validaciones 2p3