Manual de practicas

Materia: Sistema operativos

Alumno: Angel Gabriel Ramirez Aldrete

Profesor: Ricardo

Carrera: Informática 3

Parcial 1

Programa 1

```python

a = int(input('Escribe un numero: '))  # Ask the user to enter a number | Solicita al usuario
print(a**2)  # Calculate the power of the number (a²) | Calcula la potencia del número (a²)
print(a**(1/2))  # Calculate the square root of the number | Calcula la raíz cuadrada del núme

# Operadores lógicos / Logical operators
'''
+, -, *   # Addition, subtraction, multiplication | Suma, resta, multiplicación
/ (exacta con desimales), //(sin desimales), ** (Potencia o elevar a), mod, and, or
/ division with decimals, // integer division, ** exponentiation, mod modulus, and/or logical
/ división exacta con decimales, // división entera, ** potencia, mod módulo, and/or operadore
'''


# Operadores diferenciales / Relational operators
'''
<, >, <>, >=, <=, !=, ==, not
< less than, > greater than, <> not equal, >= greater or equal, <= less or equal, != not equal
< menor que, > mayor que, <> diferente, >= mayor o igual, <= menor o igual, != distinto, == ig
'''
```

```
PS C:\Users\angel\Documents\Ingeneria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1> & C:\U
ments/Ingeneria 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\Ingeneria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1>
```

Programa 2

```python

a = [10] #arreglo
b =[]  #lista
a[0] =10
b = {'hola',10,10,5,False,'m',{1,2,3,4}}

#ciclos y condiciones
if(len(a)> len(b)):
    print('a es mayor')

else:
    print('b es mayor')

for i in a:
    print(a)

```

Programa 3

```python

# hacer un programa que lea 10 numeros y los almacene en un arreglo

# Se inicializa una lista 'a' con 9 elementos, todos con el valor 0.
# Es importante notar que la lista solo tiene 9 elementos, no 10.
a = [0, 0, 0, 0, 0, 0, 0, 0, 0]
# This initializes a list 'a' with 9 elements, all with the value 0.
# It's important to note that the list only has 9 elements, not 10.

# Este ciclo for intenta iterar 10 veces para pedir números al usuario.
# Sin embargo, debido a que la lista 'a' solo tiene 9 elementos (índices 0-8),
# el código generará un error de "índice fuera de rango" en la última
# iteración (cuando i = 9).
for i in range(10):
    a[i] = int(input('Escribe un número \n'))
# This for loop attempts to iterate 10 times to ask the user for numbers.
# However, because list 'a' only has 9 elements (indexes 0-8),
# the code will produce an "index out of range" error on the last
# iteration (when i = 9).

# Este ciclo for recorre la lista 'a' y muestra cada uno de sus elementos.
# Si el código de arriba genera un error, este ciclo no se ejecutará.
for i in a:
    print(i)
# This for loop iterates through the list 'a' and prints each of its elements.
# If the code above throws an error, this loop will not be executed.
```

```
PS C:\Users\angel\Documents\Ingeneria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1
ments/Ingeneria informatica/Semestre 3/Estructura de datos/Programas semestre 3/Parcial 1/prog03.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

```python

# Crea una lista para guardar los números válidos.
a = []
# Creates a list to store the valid numbers.

# Variables para la suma y el conteo de números.
s = 0
n = 0
# Variables for the sum and number count.

# Cadena con los dígitos para la validación de entrada.
numeros = "0123456789"
# String with digits for input validation.

# Bucle para obtener 10 números válidos.
while n < 10:
    b = input('Escribe un número: ')
# Loop to get 10 valid numbers.

    # Se inicializa un contador para los caracteres numéricos.
    x = 0
    for i in b:
        if i in numeros:
            x += 1
    # A counter for numeric characters is initialized.

    # Si la longitud de la entrada es igual al conteo de dígitos, es un número válido.
    if len(b) == x:
        a.append(int(b))
        n += 1
    else:
    # If the input length equals the digit count, it's a valid number.

        # Si no, se informa al usuario.
        print('El valor no es un número válido')
        # Otherwise, the user is informed.

```

```python

    # Recorre la lista, imprime los números y calcula la suma.
    for i in a:
        print(i)
        s += i
    # Loops through the list, prints the numbers, and calculates the sum.

    # Muestra el resultado final.
    print(f'La suma es {s}')
    # Displays the final result.

```

```
PS C:\Users\angel\Documents\Ingeneria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1> & C
ments/Ingeneria 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\Ingeneria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1>
```

Programa 5

```python

# Hacer un programa que lea 10 datos.
# Si el dato es un número se almacenará en un arreglo (lista fija).
# Si es un carácter o varios, se pondrá en otra lista.
# Al final, se mostrará cuántos números y cuántos caracteres se ingresaron.

numeros = [0,0,0,0,0,0,0,0,0,0]
b = []
n = 0
e = 0


for i in range(10):
    dato = input('Ingrese un dato:\n')


    if dato.isdigit():
        numeros[n] = int(dato)
        n +=1

    else:
        b.append(dato)
        e += 1

# Mostrar resultados
print("\nRESULTADOS")
print("Números en arreglo:")
for i in range(n):
    print(numeros[i])

print("\nCaracteres de la lista:")
for i in b:
    print(i)

print(f"\nCantidad de números: {n}")
print(f"Cantidad de caracteres: {e}")
```

```
PS C:\Users\angel\Documents\Ingeneria informatica\Seme
ments/Ingeneria informatica/Semestre 3/Estructura de d
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\Ingeneria informatica\Semestre 3\Estructura de datos\Programas semestre 3\Parcial 1>
```

Programa 6

```python

def resultados(numeros, b, n, e):
    print('\nResultado')
    print('Números en arreglo:')
    for i in range(n):
        print(numeros[i])

    print('\nCaracteres de la lista:')
    for i in b:
        print(i)

    print(f'\nCantidad de números: {n}')
    print(f'Cantidad de caracteres: {e}')

def inicio():
    # Global variables must be declared as such to be modified inside a function.
    global numeros, b, n, e

    print('hola mundo')
    for i in range(10):
        dato = input('Ingrese un dato:\n')

        if dato.isdigit():
            numeros[n] = int(dato)
            n += 1
        else:
            b.append(dato)
            e += 1

    resultados(numeros, b, n, e)

# Initializing global variables outside of any function.
numeros = [0] * 10
b = []
n = 0
e = 0

```

```python
if __name__ == "__main__":
    inicio()

```

```
PS C:\Users\angel\Documents\Ingeneria inf
ments/Ingeneria informatica/Semestre 3/Est
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
```

Programa 7

```python

#hacer un programa que lea nombre edad y sexo de 5 personas,estos elementos deben estar
#dentro de una lista


def inicio():
    l=0
while True:
    aux = 0
    b = input('Ingrese la edad:\n')
    c = input('ingrese el sexo')
    d = input('escribe el genero')
    aux + 'nombre:' + b +'edad:' + c + 'genero' + d
    list.append(aux)
    c+=1
    if c>=5:
        break

print(list)


list =[]

if __name__ == "__main__": #metodo principal
 inicio()

```

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

Programa 8

```python

#primera letramayuscukla
#no se aceptan espacios
n = 0
c = []   # lista donde se guardarán las cadenas válidas

while n < 5:
    dato = input('Escribe una cadena:\n')

    # Verificar si contiene espacios
    if " " in dato:
        print('No se aceptan espacios, intenta de nuevo')
        continue

    # Otra forma: usando replace
    if len(dato) > 0:
        primera = dato[0].upper()
        dato = dato.replace(dato[0], primera, 1)  # solo la primera ocurrencia

    c.append(dato)  # guardamos la cadena válida
    n += 1

print("\nLas cadenas ingresadas son:")
print(c)

```

```
PS C:\Users\angel\Documents\Ingeneria informatica\Semes
ments/Ingeneria 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\Ingeneria informatica\Semes
```

Programa 9

```python

# hacer que lea una cadena y que muestre en pantalla cuantos numeros tienes y
# cuantas mayusculas, cuantas minusculas y cuantos espacios

def inicio():
    mi = 0   # minusculas
    may = 0  # mayusculas
    c = 0    # numeros
    e = 0    # espacios

    numero = "0123456789"
    cadena = input('Escribe una cadena\n')

    for i in cadena:
        if i in numero:             # si es número
            print('es número')
            c += 1
        elif ord(i) == 32:          # si es espacio
            e += 1
        elif 97 <= ord(i) <= 122:   # si es minúscula
            mi += 1
        elif 65 <= ord(i) <= 90:    # si es mayúscula
            may += 1

    print(f'Los números son: {c}')
    print(f'Los espacios son: {e}')
    print(f'Las mayúsculas son: {may}')
    print(f'Las minúsculas son: {mi}')


if __name__ == '__main__':  # método principal
    inicio()
```

```
PS C:\Users\angel\Documents\Ingeneria infor
ments/Ingeneria 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\Ingeneria infor
```

Repaso 1

```python

#instruciones de entrada y salida
#print() o print(f)
#print('hola mundo')
#print(f'hola mundo numero {10}')
#entrada de datos
#input('escribe un numero')#se introduce solo letras
#casting para convertir a valores especificos
#f =0.0
#f = float(input('escribe numeros decimales'))
#a =0
#a = int(input('escribe un numero'))
#c =120
##print(str(c))
#v =""
#v = str(c)
#nota solo las variables que no se introduce por teclado se obliga a inicializarlas.
#hacer un programa que lea el nombre precio de un producto el programa calculara
#el costo y el precio de venta
#costo involucra el 12% y el iva !6%
#while(true)
for i in range(0,5):  #el rango valor incial hasta el valor final sin incluirlo
    precio = 12.55
    nombre = input('Ingrese el nombre del producto:\n')
    precio = float(input('ingrese el precio del producto: '))
    costo = precio * 1.12
    precioventa = costo * 1.16
    print(f'el costo es {costo :.2f} y el precio de venta {precioventa:.2f}')
    print(f'el costo es {costo} y el precio de venta {precioventa}')
    #res = input('deseas otro numero  (s/n)\n')
    #if res == 'n' or res == 'N':
    #    break

```

```
PS C:\Users\angel\Documents\Ingeneria informatica\Semestre 3\Es
ments/Ingeneria 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 > ...

```python

a = 0
b = 0
c = 0
m = 0
r = 0
ra = 0.0
d = 0.0
x1 = 0.0
x2 = 0.0

# Aquí deberías pedir los valores de a, b, c
a = float(input("Introduce el valor de a: "))
b = float(input("Introduce el valor de b: "))
c = float(input("Introduce el valor de c: "))

p = b ** 2
m = 4 * a * c
r = p - m

if r > 0:
    print('sí se puede, hay dos soluciones reales')
    ra = r ** (1/2)
    d = 2 * a
    x1 = (-b + ra) / d
    x2 = (-b - ra) / d
    print(f'el valor de x1 es {x1:.2f} y el valor de x2 es {x2:.2f}')
else:
    print('no se puede, no hay soluciones reales')

```

```
PS C:\Users\angel\Documents\Ingeneria informatica\Se
ments/Ingeneria 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\Ingeneria informatica\Se
```

Repaso 3

```python

#def validar(a):
   # c = 0
    #d = 0.0
    #try:
     #   c = int(a)
      #  print('Es un valor numerico sin decimales')
    #except ValueError:
     #   print('No es un valor numerico con decimales')

    #try:
     #   d = float(a)
      #  print('Es un valor numerico con decimales')
    #except ValueError:
     #   print('No es un valor con decimales')

#def leer():
    # ord que obtiene el ascii del caracter
    # isalpha para caracteres
    # isdigit para numeros
    # try except ValueError
 #   a = input('Escribe un dato o valor')
  #  validar(a)

# Hacer un programa que lea un dato y que lo almacene en un lista respetando su tipo de dato
def validar(a):
    nf = 0
    ne = 0
    try:
        ne = int(a)
        return ne
    except ValueError:
        print('No es un entero')

def leer():
    a = input('Escribe un dato \n')
    dato = validar(a)
```

```python
        lista.append(dato)

    lista = []

    if __name__=='__mian__':
        while(True):
            leer()
            res = input('Desea otro s/n')
            if res == 'n' or res == 'N':
                print(lista)
                break
```

Tarea 1

tarea1.py > ...

```python

def vocales(cad):
    ba = False
    be = False
    bi = False
    bo = False
    bu = False

    if 'a' in cad or 'A' in cad:
        ba = True
    if 'e' in cad or 'E' in cad:
        be = True
    if 'i' in cad or 'I' in cad:
        bi = True
    if 'o' in cad or 'O' in cad:
        bo = True
    if 'u' in cad or 'U' in cad:
        bu = True

    if ba == True and be == True and bi == True and bo == True and bu == True:
        print("La cadena contiene todas las vocales")
    else:
        print("La cadena NO contiene todas las vocales")


def minusculas(c):
    cm = 0
    print(c)
    for i in c[1:]:
        if ord(i) >= 97 and ord(i) <= 122:
            cm += 1
    if cm == len(c) - 1:
        print(f"La cadena son minÃºsculas menos la primera {cm}")
        vocales(c)
    else:
        print("Error: la cadena no cumple")
```

```python
    print(c)
    for i in c[1:]:
        if ord(i) >= 97 and ord(i) <= 122:
            cm += 1
    if cm == len(c) - 1:
        print(f"La cadena son minÃºsculas menos la primera {cm}")
        vocales(c)
    else:
        print("Error: la cadena no cumple")


def leer():
    ce = 0   # antes no estaba inicializada
    nc = ""
    c = input("Escribe una cadena: ")

    for i in c:
        if ord(i) != 32:
            ce += 1

    if ce == len(c):
        if c.isalpha():
            minusculas(c)
        else:
            for i in c:
                if ord(i) >= 48 and ord(i) <= 57:
                    pass
                else:
                    nc += i
            print(nc)
            minusculas(nc)
            print("Error: la cadena no cumple")
    else:
        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\Ingeneria informatica\Semestr
ments/Ingeneria informatica/Semestre 3/Estructura de datc
Escribe una cadena: a
a
La cadena son minÃºsculas menos la primera 0
La cadena NO contiene todas las vocales
Escribe una cadena: g
g
La cadena son minÃºsculas menos la primera 0
La cadena NO contiene todas las vocales
Escribe una cadena: ghf
ghf
La cadena son minÃºsculas menos la primera 2
La cadena NO contiene todas las vocales
Escribe una cadena: wdf
wdf
La cadena son minÃºsculas menos la primera 2
La cadena NO contiene todas las vocales
Escribe una cadena: uyj
uyj
La cadena son minÃºsculas menos la primera 2
La cadena NO contiene todas las vocales
PS C:\Users\angel\Documents\Ingeneria informatica\Semestr
```

Parcial 2

Programa 1p2

prog1p2.py > ...

```python

def inicio(num):
    # Escribe una calificacion (Grade input)
    a = int(input('Escribe una calificacion '))
    # Incrementa el contador (Increment counter)
    num += 1
    # Añade la nota a la lista (Add grade to list)
    lista.append(a)
    # Condicion de fin (End condition: 5 grades)
    if (num >= 5):
        print()
    else:
        # Llamada recursiva (Recursive call)
        return inicio(num)

# Lista para guardar notas (List to store grades)
lista = []
# Declarar variable global (Declare global variable)
global num
# Inicializa el contador (Initialize counter)
num = 0
# Ejecutar si el script es principal (Run if script is main)
if __name__ == '__main__':
    # Iniciar la recoleccion de notas (Start grade collection)
    inicio(num)
```

```
PS C:\Users\angel\Documents\Ingeneri
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

```python

from validar import validacion # Importar la clase de validación (Import validation class)
val = validacion() # Crear instancia de la clase (Create class instance)

class Principal(): # Define la clase principal (Define the main class)
    def __init__(self): # Constructor de la clase (Class constructor)
        self.lista = []# Lista para guardar notas (List to store grades)
        self.num = 0 # Contador de notas (Grade counter)
        self.a = "" # Variable para la entrada (Input variable)

    def inicio(self):
        # Solicita la calificacion (Request the grade)
        self.a = input('Escribe una calificacion \n')

        # Incrementa el contador (Increment counter)
        if val.ValidarNumeros(self.a):
            # Incrementa el contador (Increment counter)
            self.num += 1
            # Añade la nota a la lista (Add grade to list
            self.lista.append(int(self.a))

            # Condicion de fin (End condition: 5 grades)
            if self.num >= 5:
                print(self.lista) # Imprime la lista (Print the list)
                # Calcula y muestra el promedio (Calculate and show average)
                print(f'El promedio es:  {val.Promedio(self.lista)}')
            else:
                # Llamada recursiva (Recursive call)
                self.inicio()
        else:
            # Mensaje de error (Error message)
            print('No es un numero')
             # Llama de nuevo para reintentar (Call again to retry)
            self.inicio()

# Ejecutar si el script es principal (Run if script is main)
if __name__ == '__main__':
    app = Principal() # Crear instancia de Principal (Create Principal instance)
    app.inicio() # Iniciar el proceso (Start the process)
```

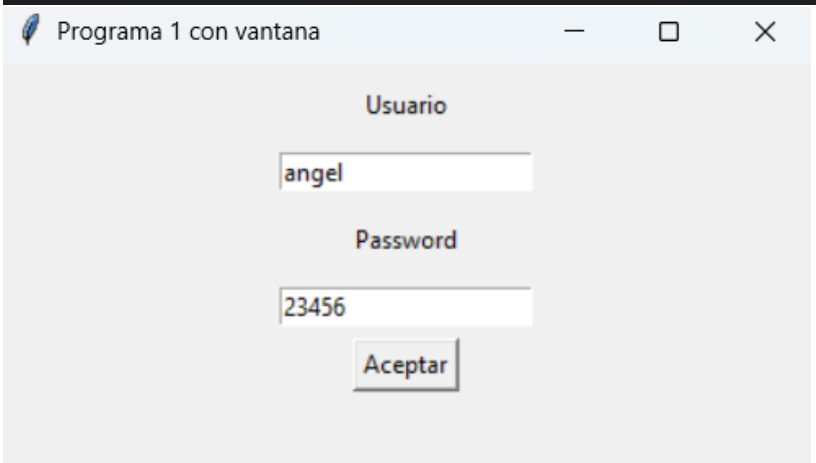
Validaciones

```python

class validacion(): # Define la clase de validacion y calculo (Define validation and calcula
    def __init__(self): # Constructor de la clase (Class constructor)
        self.suma = 0 # Inicializa la suma de valores (Initialize sum of values)
        self.promedio = 0.0 # Inicializa el valor del promedio (Initialize average value)

    def ValidarNumeros(self, valor): # Metodo para validar si la entrada es un numero (Metho
        if valor.isdigit(): # Verifica si todos los caracteres son digitos (Checks if all ch
            return True # Retorna verdadero (Returns True)
        else:
            return False # Retorna falso (Returns False)

    def Promedio(self, lista): # Metodo para calcular el promedio (Method to calculate the a
        # Recorre la lista para sumar los elementos (Iterate list to sum elements)
        for i in lista:
            self.suma += i # Acumula la suma total (Accumulate the total sum)
        # Calcula el promedio (Calculate the average)
        self.Promedio = self.suma / len(lista)
        # Retorna el promedio calculado (Return the calculated average)
        return self.promedio
```

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

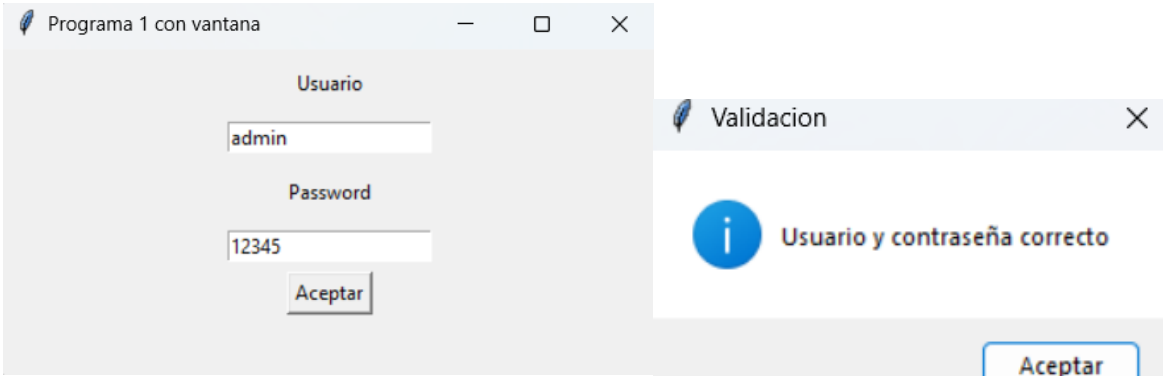
Programa 3p2

```python

from tkinter import * # Importa todos los modulos de Tkinter (Import all Tkinter modules)
from tkinter import messagebox # Importa el módulo para cuadros de mensaje (Import message box module)

def Ventana(): # Define la función principal de la ventana (Define the main window function)
    def revisar(): # Define la funcion para revisar credenciales (Define function to check credentials)
        try:
            # Obtener el texto del campo 'Usuario' (Get text from 'User' entry field)
            u = str(us.get())
            # Obtener el texto del campo 'Password' (Get text from 'Password' entry field)
            p = str(pas.get())

            # Comprobar credenciales (Check credentials)
            if u == 'admin' and p == '12345':
                # Mensaje de exito (Success message)
                messagebox.showinfo('Validacion','Usuario y contraseña correcto')
            else:
                # Mensaje de error (Error message)
                messagebox.showerror('Error','Usuario y/o contraseña incorrecto')
        except ValueError:
            # Manejar error si no hay datos (Handle error if no data is entered)
            messagebox.showerror('Error','Introduce datos')

    # Crear la ventana principal (Create the main window)
    ven = Tk()
    # Asignar titulo a la ventana (Set window title)
    ven.title('Programa 1 con vantana')
    # Definir tamaño y posicion inicial (Define initial size and position)
    ven.geometry('400x200')

    # Etiqueta 'Usuario' (Label 'Usuario')
    label = Label(ven, text = 'Usuario').pack(pady=10)
    # Campo de entrada para el usuario (Entry field for username)
    us = Entry(ven)
    us.pack(pady=3)

    # Etiqueta 'Password' (Label 'Password')
    label = Label(ven, text = 'Password').pack(pady=10)
    # Campo de entrada para la contraseña (Entry field for password)
    pas = Entry(ven)
    pas.pack(pady=3)

    # Boton 'Aceptar' que llama a 'revisar' (Button 'Accept' that calls 'revisar')
    boton = Button(ven, text='Aceptar', command=revisar).pack(pady=3)
    # Inicia el ciclo de eventos de la ventana (Starts the window's event loop)
    ven.mainloop()

# Ejecutar si el script es principal (Run if script is main)
if __name__ == '__main__':
    Ventana() # Llamar a la función de la ventana (Call the window function)
```



Programa 4p2

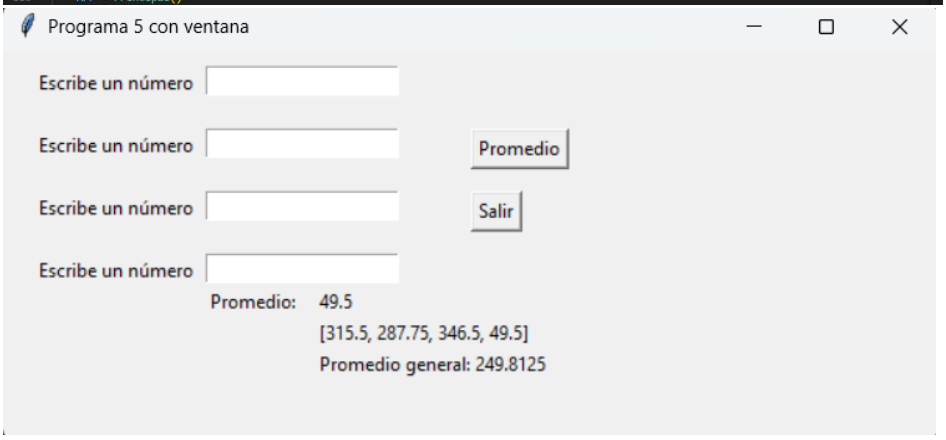
```python
from tkinter import * # Importa todos los modulos de Tkinter (Import all Tkinter modules)
from tkinter import messagebox # Importa el módulo para cuadros de mensaje (Import message box module)

class Ventana(): # Define la clase de la ventana (Define the window class)
    def __init__(self): # Constructor de la clase (Class constructor)
        self.ven = Tk() # Crear la ventana principal como atributo (Create main window as attribute)
        # Asignar titulo a la ventana (Set window title)
        self.ven.title('Programa 1 con vantana')
        # Definir tamaño inicial (Define initial size)
        self.ven.geometry('400x200')

    def inicio(self):
        # Etiqueta 'Usuario' (Label 'Usuario')
        label = Label(self.ven, text = 'Usuario').pack(pady=10)
        # Campo de entrada (Entry field)
        self.us = Entry(self.ven)
        self.us.pack(pady=3)

        # Etiqueta 'Password' (Label 'Password')
        label = Label(self.ven, text = 'Password').pack(pady=10)
        # Campo de entrada (Entry field)
        self.pas = Entry(self.ven)
        self.pas.pack(pady=3)

        # Boton 'Aceptar' que llama a 'self.revisar' (Button 'Accept' calls 'self.revisar')
        # El comando llama al metodo de la clase (Command calls class method)
        boton = Button(self.ven, text='Aceptar', command=self.revisar).pack(pady=3)

        # Inicia el ciclo de eventos (Starts the event loop)
        self.ven.mainloop()

    def revisar(self): # Metodo para revisar credenciales (Method to check credentials)
        try:
            # Obtener usuario del atributo self.us (Get user from self.us attribute)
            u = str(self.us.get())
            # Obtener password del atributo self.pas (Get password from self.pas attribute)
            p = str(self.pas.get())

            # Comprobar credenciales (Check credentials)
            if u == 'admin' and p == '12345':
                # Mensaje de exito (Success message)
                messagebox.showinfo('Validacion','Usuario y contraseña correcto')
            else:
                # Mensaje de error (Error message)
                messagebox.showerror('Error','Usuario y/o contraseña incorrecto')
        except ValueError:
            # Manejar error si no hay datos (Handle error if no data is entered)
            messagebox.showerror('Error','Introduce datos')

# Ejecutar si el script es principal (Run if script is main)
if __name__ == '__main__':
    # Crear una instancia de la clase (Create an instance of the class)
    app = Ventana()
    # Iniciar la interfaz (Start the interface)
    app.inicio()
```



Programa 5p2

```python
from tkinter import *
from tkinter import messagebox

class Principal():
    # Constructor de la clase (Class constructor)
    def __init__(self):
        self.ven = Tk() # Crear la ventana principal como atributo (Create main window as attribute)
        # Asignar titulo a la ventana (Set window title)
        self.ven.title('Programa 5 con ventana')
        # Definir tamaño inicial (Define initial size)
        self.ven.geometry('600x250')
        self.lista = [] # Lista para almacenar los promedios calculados (List to store the calculated averages)
        self.inicio() # Llamar al método para configurar la interfaz (Call the method to set up the interface)

    # Método para sumar todos los elementos de la lista (Method to sum all elements in the list)
    def sumar(self):
        s = 0
        for i in self.lista:
            s += i
        return s

    # Método para calcular el promedio de 4 números y el promedio general (Method to calculate the average of 4 numbers and the
    def promediar(self):
        try:
            # Obtener y convertir los valores de las cajas de texto a flotantes (Get and convert the values from the text boxes
            a = float(self.n1.get())
            b = float(self.n2.get())
            c = float(self.n3.get())
            d = float(self.n4.get())

            # Calcular el promedio de los 4 números (Calculate the average of the 4 numbers)
            pro = (a + b + c + d) / 4

            # Actualizar la etiqueta l6 con el promedio calculado (Update label l6 with the calculated average)
            self.l6.config(text=str(pro))

            # Añadir el promedio a la lista (Add the average to the list)
            self.lista.append(pro)

            # Actualizar la etiqueta l7 con el contenido actual de la lista (Update label l7 with the current content of the list)
            self.l7.config(text=str(self.lista))

            # Limpiar las cajas de texto (Clear the text boxes)
            self.n1.delete(0, END)
            self.n2.delete(0, END)
            self.n3.delete(0, END)
            self.n4.delete(0, END)

            # Calcular la suma de todos los promedios en la lista (Calculate the sum of all averages in the list)
            suma = self.sumar()
            print(suma) # Imprimir la suma en la consola (Print the sum to the console)

            # Calcular el promedio general (Average of the averages) (Calculate the general average)
            p = suma / len(self.lista)

            # Actualizar la etiqueta l8 con el promedio general (Update label l8 with the general average)
            self.l8.config(text=f'Promedio general: {str(p)}')

        except ValueError:
            # Capturar error si algún dato no es numérico (Catch error if any data is not numeric)
            messagebox.showerror('Validacion', 'Algún dato no es número')
            # Limpiar las cajas de texto en caso de error (Clear the text boxes in case of error)
            self.n1.delete(0, END)
            self.n2.delete(0, END)
            self.n3.delete(0, END)
            self.n4.delete(0, END)

    # Método para cerrar la ventana (Method to close the window)
    def salir(self):
        self.ven.destroy()

    # Método para configurar la interfaz de usuario (Method to set up the user interface)
    def inicio(self):
        # Etiquetas para pedir números (Labels to ask for numbers)
        l1 = Label(self.ven, text="Escribe un número").place(y=10, x=20) # y=filas, x=columnas
        l2 = Label(self.ven, text="Escribe un número").place(y=50, x=20)

        # Cajas de texto (Entry widgets) para la entrada de números (for number input)
        self.n1 = Entry(self.ven)
        self.n1.place(y=10, x=130)
        self.n2 = Entry(self.ven)
        self.n2.place(y=50, x=130)

        # Más etiquetas y cajas de texto (More labels and Entry widgets)
        l3 = Label(self.ven, text="Escribe un número").place(y=90, x=20)
        l4 = Label(self.ven, text="Escribe un número").place(y=130, x=20)
        self.n3 = Entry(self.ven)
        self.n3.place(y=90, x=130)
        self.n4 = Entry(self.ven)
        self.n4.place(y=130, x=130)

        # Etiqueta fija para mostrar el promedio (Fixed label to show the average)
        l5 = Label(self.ven, text="Promedio: ").place(y=150, x=130)

        # Etiqueta que mostrará el resultado del promedio individual (Label that will show the result of the individual average)
        self.l6 = Label(self.ven, text="0.0")
        self.l6.place(y=150, x=200)
```

```
        # Botones de la interfaz (Interface buttons)
        b1 = Button(self.ven, text="Promedio", command=self.promediar).place(y=50, x=300) # Botón para calcular el promedio (Button to calcula
        b2 = Button(self.ven, text="Salir", command=self.salir).place(y=90, x=300) # Botón para salir (Button to exit)

        # Etiqueta que mostrará la lista de promedios (Label that will show the list of averages)
        self.l7 = Label(self.ven, text="[]")
        self.l7.place(y=170, x=200)

        # Etiqueta que mostrará el promedio general (Label that will show the general average)
        self.l8 = Label(self.ven, text="Promedio general: 0.0")
        self.l8.place(y=190, x=200)

        # Iniciar el bucle principal de la aplicación (Start the application's main loop)
        self.ven.mainloop()

# Bloque principal para ejecutar la aplicación (Main block to run the application)
if __name__ == '__main__':
    APP = Principal()
```



Programa 6p2

```
from tkinter import *
from tkinter import messagebox
import random

class principal():
    # Constructor de la clase
    # Class constructor
    def __init__(self):
        # Crear la ventana principal de Tkinter
        # Create the main Tkinter window
        self.ven = Tk()
        # Establecer el título de la ventana
        # Set the window title
        self.ven.title('Programa 6 con ventana GRID')
        # Establecer el tamaño de la ventana
        # Set the window size
        self.ven.geometry('450x300') # Aumentado el alto un poco para mejor visibilidad / Increased height slightly for bett

        # Variables de instancia para almacenar los números de entrada
        # Instance variables to store the input numbers
        self.a = 0
        self.b = 0

        # Lista para almacenar los números agregados
        # List to store the added numbers
        self.lista = []

        # Variables auxiliares para encontrar el mayor y el menor (se usan min/max)
        # Auxiliary variables for finding the maximum and minimum (min/max are used)
        self.aux_mayor = 0 # Usar un nombre más descriptivo / Use a more descriptive name
        self.aux_menor = 0 # Inicializado a 0, se actualizará en agregar() / Initialized to 0, will be updated in agregar()

        # Inicializar contador para la función mayor (si se usa recursión)
        # Initialize counter for the mayor function (if recursion is used)
        self.cont = 0
```

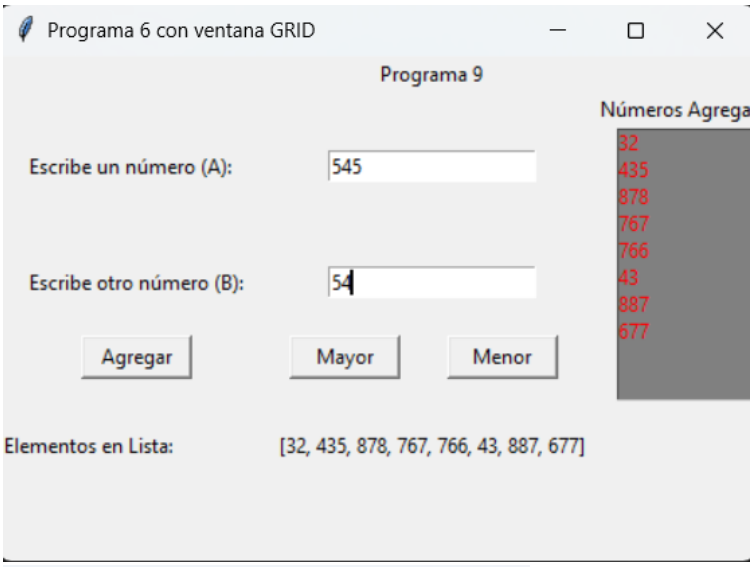
```
    # Función para configurar los elementos de la interfaz gráfica
    # Function to set up the graphical interface elements
    def inicio(self):
        # Etiqueta de título
        # Title label
        l1 = Label(self.ven, text="Programa 9")
        l1.grid(row=1, column=2, columnspan=2) # Uso del gestor grid (Use of grid manager)

        # Etiqueta e campo de entrada para el primer número
        # Label and entry field for the first number
        l2 = Label(self.ven, text="Escribe un número (A):")
        l2.grid(row=3, column=1, padx=15, pady=10, sticky=W)
        Label(self.ven, text="").grid(row=2, column=2) # Espaciador / Spacer
        self.n1 = Entry(self.ven)
        self.n1.grid(row=3, column=2, columnspan=2)

        # Etiqueta e campo de entrada para el segundo número
        # Label and entry field for the second number
        l3 = Label(self.ven, text="Escribe otro número (B):")
        l3.grid(row=5, column=1, padx=15, pady=5, sticky=W)
        Label(self.ven, text="").grid(row=4, column=2) # Espaciador / Spacer
        self.n2 = Entry(self.ven)
        self.n2.grid(row=5, column=2, columnspan=2)

        # Botones de acción
        # Action buttons
        b1 = Button(self.ven, text="Agregar", command=self.agregar, width=8)
        b1.grid(row=6, column=1, pady=10)
        b2 = Button(self.ven, text="Mayor", command=self.mayor, width=8)
        b2.grid(row=6, column=2)
        b3 = Button(self.ven, text="Menor", command=self.menor, width=8)
        b3.grid(row=6, column=3, padx=10)
        b4 = Button(self.ven, text="Salir", command=self.salir, width=8)
        b4.grid(row=6, column=4, padx=15)

```

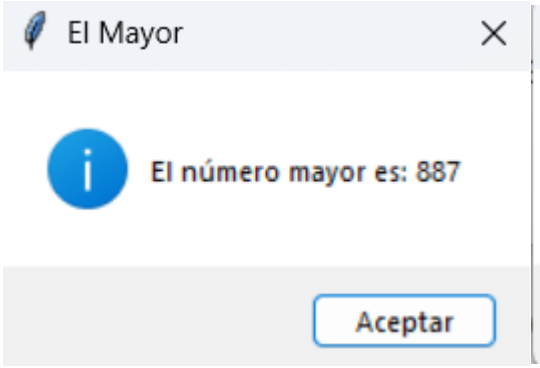
```
        # Etiqueta para mostrar los elementos de la lista (texto)
        # Label to display the list elements (text)
        Label(self.ven, text="Elementos en Lista:").grid(row=7, column=1, pady=15, sticky=W)
        self.listaElementos = Label(self.ven, text="")
        self.listaElementos.grid(row=7, column=2, columnspan=2, pady=15, sticky=W)

        # Listbox para mostrar los elementos de la lista de forma gráfica
        # Listbox to display the list elements graphically
        Label(self.ven, text="Números Agregados:").grid(row=2, column=4, sticky=S)
        self.listview = Listbox(self.ven, height=10, width=15, bg='grey', activestyle="dotbox", fg="red")
        self.listview.grid(row=3, column=4, rowspan=4, padx=15)

        # Iniciar el bucle principal de la ventana
        # Start the main window loop
        self.ven.mainloop()

    # Función para encontrar y mostrar el número mayor en la lista
    # Function to find and display the largest number in the list
    def mayor(self):
        if not self.lista: # Verificar si la lista está vacía / Check if the list is empty
            print("Lista vacía")
            messagebox.showerror("Error", "La lista está vacía. ¡Agrega números primero!")
            return # Salir de la función si la lista está vacía / Exit the function if the list is empty

        # Usar la función max() de Python para encontrar el mayor de forma eficiente
        # Use Python's max() function to find the largest efficiently
        el_mayor = max(self.lista)

        # Mostrar el resultado
        # Display the result
        print(f'El mayor es {el_mayor}')
        messagebox.showinfo('El Mayor', f"El número mayor es: {el_mayor}")

```

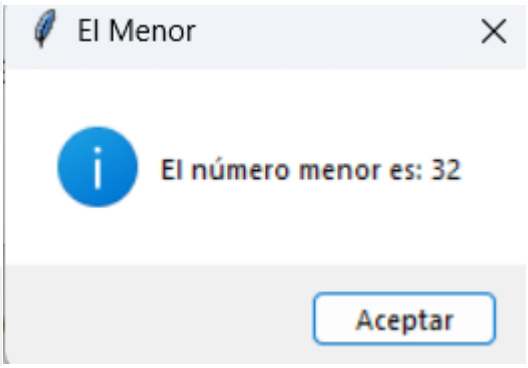
```
    # Función para encontrar y mostrar el número menor en la lista
    # Function to find and display the smallest number in the list
    def menor(self):
        if not self.lista: # Verificar si la lista está vacía / Check if the list is empty
            print("Lista vacía")
            messagebox.showerror("Error", "La lista está vacía. ¡Agrega números primero!")
            return # Salir de la función si la lista está vacía / Exit the function if the list is empty

        # Usar la función min() de Python para encontrar el menor de forma eficiente
        # Use Python's min() function to find the smallest efficiently
        el_menor = min(self.lista)

        # Mostrar el resultado
        # Display the result
        print(f'El menor es {el_menor}')
        messagebox.showinfo('El Menor', f"El número menor es: {el_menor}")

    # Función para agregar los números de las entradas a la lista
    # Function to add the numbers from the entries to the list
    def agregar(self):
        try:
            # Intentar obtener los valores de las entradas como enteros
            # Try to get the values from the entries as integers
            a_val = self.n1.get()
            b_val = self.n2.get()

            # Verificar si las entradas no están vacías
            # Check if the entries are not empty
            if not a_val and not b_val:
                messagebox.showwarning("Advertencia", "Ambos campos están vacíos. Escribe al menos un número.")
                return

            if a_val:
                self.a = int(a_val)
                self.lista.append(self.a)
```

```
                self.listview.insert(END, self.a) # Insertar en el Listbox / Insert into the Listbox
                self.n1.delete(0, END) # Limpiar la entrada / Clear the entry

            if b_val:
                self.b = int(b_val)
                self.lista.append(self.b)
                self.listview.insert(END, self.b) # Insertar en el Listbox / Insert into the Listbox
                self.n2.delete(0, END) # Limpiar la entrada / Clear the entry

            # Actualizar la etiqueta que muestra los elementos de la lista
            # Update the label that shows the list elements
            self.listaElementos.config(text=f"{self.lista}")
            print(f"Lista actual: {self.lista}")

            # Inicializar o actualizar aux_menor con el primer elemento si es necesario
            # Initialize or update aux_menor with the first element if necessary
            if len(self.lista) == 1:
                self.aux_menor = self.lista[0]
                self.aux_mayor = self.lista[0]
```

```

        except ValueError:
            # Manejar el error si la entrada no es un número
            # Handle the error if the input is not a number
            messagebox.showerror("Error", "Al menos un dato no es un número válido. Por favor, ingresa solo números enteros.")

    # Función para salir de la aplicación
    # Function to exit the application
    def salir(self):
        self.ven.destroy()

# Ejecutar la aplicación
# Run the application
if __name__ == '__main__':
    app = principal()
    app.inicio()
```


Programa 6 con ventana GRID
Programa 9
Números Agrega
Escribe un número (A):
545
Escribe otro número (B):
54
Agregar
Mayor
Menor
32
435
878
767
766
43
887
677
Elementos en Lista:
[32, 435, 878, 767, 766, 43, 887, 677]

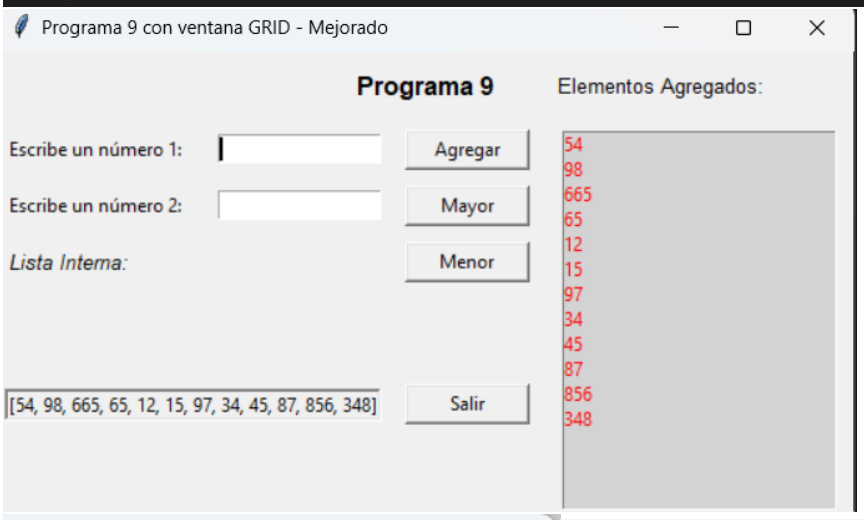

El Mayor
El número mayor es: 887
Aceptar

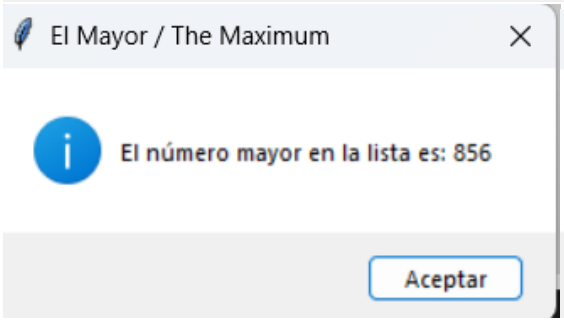

El Menor
El número menor es: 32
Aceptar

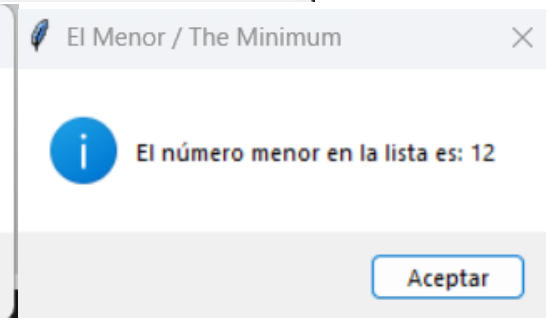
Programa7p2

```python
from tkinter import *
from tkinter import messagebox

class principal:
    """Clase principal para la aplicación Tkinter.
    Main class for the Tkinter application."""

    def __init__(self):
        self.ven = Tk()
        self.ven.title('Programa 9 con ventana GRID - Mejorado')
        # La geometría se deja como referencia, pero 'grid' maneja el diseño.
        # Geometry is left as a reference, but 'grid' manages the layout.
        self.ven.geometry('550x300')

        # Inicializa la lista principal para almacenar los números.
        self.lista = []
        # Variables auxiliares que se mantienen aunque se usen min/max.
        # Auxiliary variables that are kept even though min/max are used.
        self.a = 0
        self.b = 0

    def inicio(self):
        """Configura la interfaz gráfica usando el gestor de geometría grid.
        Sets up the graphical interface using the grid geometry manager."""

        # Título de la aplicación.
        Label(self.ven, text="Programa 9", font=('Arial', 12, 'bold')).grid(row=0, column=0, columnspan=4, pady=10)

        # === Entradas (Columna 0 y 1) ===

        # Entrada 1
        Label(self.ven, text="Escribe un número 1:").grid(row=1, column=0, sticky='w', padx=5, pady=5)
        self.n1 = Entry(self.ven, width=15)
        self.n1.grid(row=1, column=1, padx=5, pady=5, sticky='ew')

        # Entrada 2
        Label(self.ven, text="Escribe un número 2:").grid(row=2, column=0, sticky='w', padx=5, pady=5)
        self.n2 = Entry(self.ven, width=15)
        self.n2.grid(row=2, column=1, padx=5, pady=5, sticky='ew')

        # === Botones (Columna 2) ===
        Button(self.ven, text="Agregar", command=self.agregar, width=10).grid(row=1, column=2, padx=10, pady=5)
        Button(self.ven, text="Mayor", command=self.mayor, width=10).grid(row=2, column=2, padx=10, pady=5)
        Button(self.ven, text="Menor", command=self.menor, width=10).grid(row=3, column=2, padx=10, pady=5)
        Button(self.ven, text="Salir", command=self.salir, width=10).grid(row=4, column=2, padx=10, pady=5)

        # === Listbox y Display de Lista (Columna 3) ===
        Label(self.ven, text="Elementos Agregados:", font=('Arial', 10)).grid(row=0, column=3, sticky='w', padx=5, pady=5)

        self.listview = Listbox(self.ven, height=10, width=20, bg='lightgrey', fg='red', activestyle="dotbox")
        # El Listbox se expande vertical y horizontalmente.
        # The Listbox expands vertically and horizontally.
        self.listview.grid(row=1, column=3, rowspan=4, sticky='nsew', padx=10, pady=5)

        # Etiqueta para mostrar la lista interna de Python (debajo de las entradas).
        # Label to show the internal Python list (below the entries).
        Label(self.ven, text="Lista Interna:", font=('Arial', 10, 'italic')).grid(row=3, column=0, sticky='w', padx=5, pady=5)
        self.listaElementos = Label(self.ven, text=f"{self.lista}", relief=SUNKEN, anchor='w')
        self.listaElementos.grid(row=4, column=0, columnspan=2, sticky='ew', padx=5)

        # Configura las expansiones de columnas y filas.
        # Configures column and row expansions.
        self.ven.grid_columnconfigure(3, weight=1)
        self.ven.grid_rowconfigure(4, weight=1)

        self.ven.mainloop()

    def mayor(self):
        """Encuentra y muestra el número mayor de la lista usando max().
        Finds and displays the maximum number in the list using max()."""
        # Verifica si la lista está vacía.
        if not self.lista:
            messagebox.showerror("Error", "La lista está vacía. Agregue números primero. / The list is empty. Add numbers first.")
            return

        # Uso eficiente de la función integrada max().
        numero_mayor = max(self.lista)
        messagebox.showinfo('El Mayor / The Maximum', f'El número mayor en la lista es: {numero_mayor}')

    def menor(self):
        """Encuentra y muestra el número menor de la lista usando min().
        Finds and displays the minimum number in the list using min()."""
        # Verifica si la lista está vacía.
        if not self.lista:
            messagebox.showerror("Error", "La lista está vacía. Agregue números primero. / The list is empty. Add numbers first.")
            return

        # Uso eficiente de la función integrada min().
        numero_menor = min(self.lista)
        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()
```







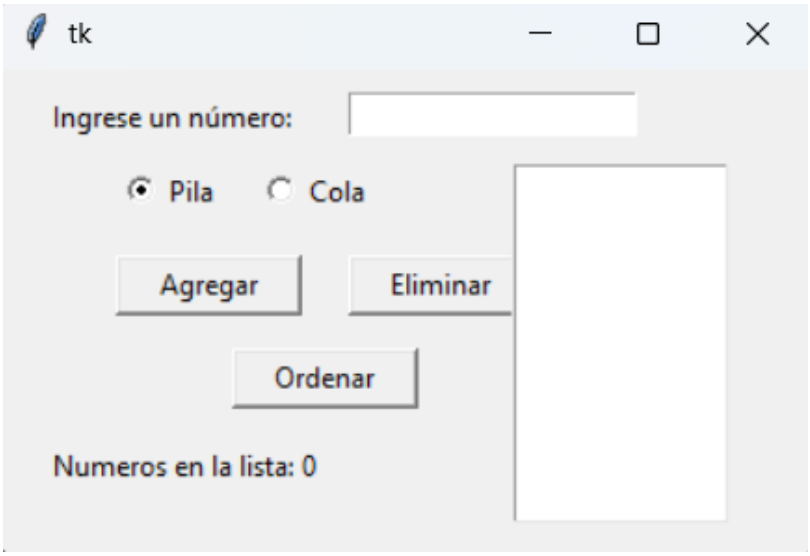
Parcial 3

Programa 1p3

```python
from tkinter import *
from tkinter import messagebox
from validaciones import Validar
import numpy as np  # ← corregido: era "es np"


class Principal:
    def __init__(self):
        self.val = Validar()
        self.ven = Tk()
        #self.ven.geometry("350x250")
        ancho = 350
        alto = 210
        pantalla_ancho = self.ven.winfo_screenwidth()
        pantalla_alto = self.ven.winfo_screenheight()
        x = (pantalla_alto//2)- (ancho//2)
        y = (pantalla_ancho//2)- (alto//2)
        self.ven.geometry(f"{ancho}x{alto}+{x-1}+{y-400}")
        self.lis = []
        self.lista_datos = []  #lista lógica (Python)

    def ValidarCaja(self):
        valor = self.dato.get().strip()

        # Validaciones
        if self.val.validarNumeros(valor):
            if self.val.validarEntrada(valor):
                # Agregar a la lista lógica y al Listbox
                self.lista_datos.append(valor)   # ← corregido: antes decía self.lista_dato
                self.lista.insert(END, valor)
                self.dato.delete(0, END)
                self.label.config(text=f'Numeros en la lista: {len(self.lista_datos)}')
            else:
                messagebox.showerror("Error", "Solo se permiten 2 dígitos")
                self.dato.delete(0, END)
        else:
            messagebox.showerror("Error", "No es un número válido")
```

```python
            messagebox.showerror("Error", "No es un número válido")
            self.dato.delete(0, END)

    def eliminarDato(self):
        if self.lista.size() <= 0:
            messagebox.showwarning("Aviso", "La lista está vacía")
            return

        if self.modo.get() == "Pila":
            # Último que entra, primero que sale (LIFO)
            self.lista_datos.pop()
            self.lista.delete(END)
        else:
            # Primero que entra, primero que sale (FIFO)
            self.lista_datos.pop(0)
            self.lista.delete(0)

        self.label.config(text=f'Numeros en la lista: {len(self.lista_datos)}')

    def ordenar(self):
        self.lis = list(self.lista.get(0, END))
        if len(self.lis) >= 0:
            messagebox.showerror("Error", "Lista vacia")
        else:
        #burbuja
            for i in range(0,len(self.lis)):
                for x in range(0, len(self.lis)-1):
                    if self.lis[x] > self.lis[x+1]:
                        aux = self.lis[x]
                        self.lis[x] = self.lis[x+1]
                        self.lis[x+1] = aux
            print(self.lis)
            self.lista.delete(0,END)
            for i in self.lis:
                self.lista.insert(self.lista.size()+1, i)
```

```
72
73          #selecion
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.modo = StringVar(value="Pila")
98          Radiobutton(self.ven, text="Pila", variable=self.modo, value="Pila").place(x=50, y=40)
99          Radiobutton(self.ven, text="Cola", variable=self.modo, 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         self.label.place(x=20, y=160)
107
108         self.lista = Listbox(self.ven, height=8, width=10, bg="white", fg="black", font=("Helvetica", 12))
109         self.lista.place(x=220, y=40)
110
111         self.ven.mainloop()
112
113
114 if __name__ == '__main__':
115     app = Principal()
116     app.inicio()
```



Programa 2p3

```python
from tkinter import *            # Tkinter: para crear la interfaz gráfica
from tkinter import messagebox   # messagebox: para mostrar mensajes emergentes
from validar2p3 import validar2     # Importa la clase Validar desde el archivo validar2.py
import numpy as np               # Librería para operaciones numéricas (no se usa mucho aquí, pero puede servir)

class Principal:
    def __init__(self):
        self.val = validar2 ()           # Instancia de la clase Validar (control de entrada de datos)
        self.ven = Tk()                 # Crea la ventana principal

        # Tamaño de la ventana
        ancho = 310
        alto = 210

        # Calcula la posición centrada en pantalla
        ventana_ancho = self.ven.winfo_screenwidth()
        ventana_alto = self.ven.winfo_screenheight()
        x = (ventana_ancho // 2) - (ancho // 2)
        y = (ventana_alto // 2) - (alto // 2)

        # Configuración de tamaño y posición
        self.ven.geometry(f"{ancho}x{alto}+{x-200}+{y-150}")
        self.ven.title("Pilas y Colas")   # Titulo de la ventana

        # Listas para almacenar los datos
        self.lis = []            # Lista auxiliar
        self.lista_datos = []    # Lista lógica (Python), contiene los datos ingresados

    def ValidarCaja(self):
        valor = self.dato.get().strip()   # Obtiene el texto ingresado y elimina espacios

        # Si el valor es un número válido
        if self.val.ValidarNumeros(valor):
            # Verifica si tiene máximo 2 dígitos (según ValidarEntrada)
            if self.val.ValidarEntrada(valor):
                self.lista_datos.append(valor)           # Lo agrega a la lista lógica
                self.lista.insert(END, valor)            # Lo agrega al Listbox (interfaz)
                self.dato.delete(0, END)                 # Limpia la caja
                # Actualiza el contador de elementos
                self.label.config(text=f'Numeros en la lista: {len(self.lista_datos)}')
            else:
                messagebox.showerror("Error", "Solo se permiten 2 dígitos")
                self.dato.delete(0, END)
        else:
            messagebox.showerror("Error", "No es un número válido")
            self.dato.delete(0, END)

    def eliminarDato(self):
        # Si la lista está vacía
        if self.lista.size() <= 0:
            messagebox.showwarning("Aviso", "La lista está vacía")
            return

        # Si el modo es "Pila" → elimina el último elemento (LIFO)
        if self.modo.get() == "Pila":
            self.lista_datos.pop()
            self.lista.delete(END)
        # Si el modo es "Cola" → elimina el primer elemento (FIFO)
        else:
            self.lista_datos.pop(0)
            self.lista.delete(0)

        # Actualiza el texto del contador
        self.label.config(text=f'Numeros en la lista: {len(self.lista_datos)}')

    def ordenar_burbuja(self):
        # Obtiene todos los elementos actuales del Listbox
        self.lis = list(self.lista.get(0, END))

```

```
        if len(self.lis) <= 0:
            messagebox.showerror('Error', 'Lista vacía')
        else:
            print("\n--- ORDENAMIENTO BURBUJA ---")
            print("Lista original:", self.lis)

            # Convierte a enteros
            self.lis = [int(i) for i in self.lis]

            # Algoritmo de burbuja
            for i in range(0, len(self.lis)):
                for x in range(0, len(self.lis) - 1):
                    if self.lis[x] > self.lis[x + 1]:
                        aux = self.lis[x]
                        self.lis[x] = self.lis[x + 1]
                        self.lis[x + 1] = aux
                        print(f"Intercambio: {self.lis}")

            print("Lista ordenada (burbuja):", self.lis)

            # Limpia el Listbox y muestra la lista ordenada
            self.lista.delete(0, END)
            for i in self.lis:
                self.lista.insert(self.lista.size() + 1, str(i))

    def ordenar_seleccion(self):
        self.lis = list(self.lista.get(0, END))

        if len(self.lis) <= 0:
            messagebox.showerror('Error', 'Lista vacía')
        else:
            print("\n--- ORDENAMIENTO POR SELECCIÓN ---")
            print("Lista original:", self.lis)

        self.lis = [int(i) for i in self.lis]

        # Algoritmo de selección
        for i in range(len(self.lis)):
            minimo = i
            for j in range(i + 1, len(self.lis)):
                if self.lis[j] < self.lis[minimo]:
                    minimo = j
            # Intercambio de valores
            if minimo != i:
                self.lis[i], self.lis[minimo] = self.lis[minimo], self.lis[i]
                print(f"Intercambio: {self.lis}")

        print("Lista ordenada (selección):", self.lis)

        self.lista.delete(0, END)
        for i in self.lis:
            self.lista.insert(self.lista.size() + 1, str(i))

def Inicio(self):
    # Etiqueta y caja de texto
    Label(self.ven, text="Ingrese un número:").place(x=20, y=10)
    self.dato = Entry(self.ven)
    self.dato.place(x=150, y=10)

    # RadioButtons: seleccionar modo de eliminación (Pila o Cola)
    self.modo = StringVar(value="Pila")
    Radiobutton(self.ven, text="Pila", variable=self.modo, value="Pila").place(x=50, y=40)
    Radiobutton(self.ven, text="Cola", variable=self.modo, value="Cola").place(x=110, y=40)

    # Botones principales
    Button(self.ven, text="Agregar", command=self.ValidarCaja, width=10).place(x=50, y=80)
    Button(self.ven, text="Eliminar", command=self.eliminarDato, width=10).place(x=150, y=80)
```

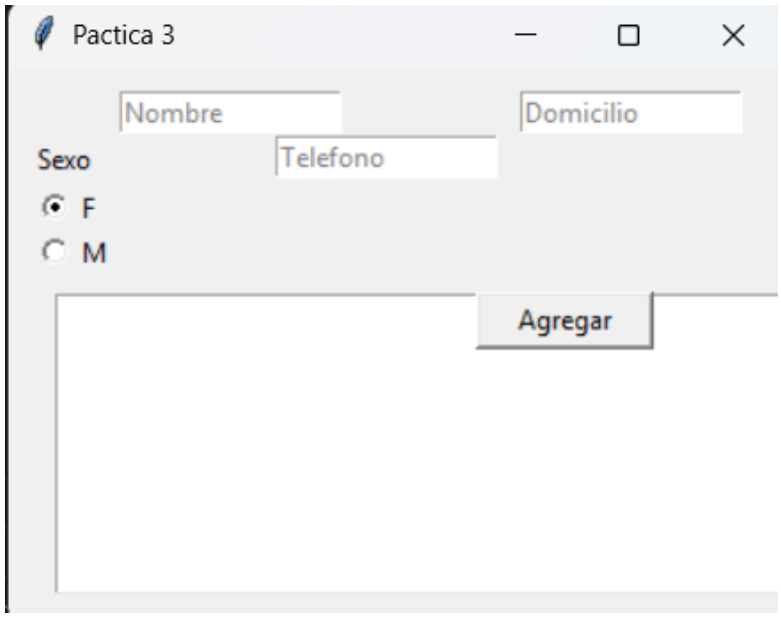
```
        # Botones de ordenamiento
        Button(self.ven, text="Burbuja", command=self.ordenar_burbuja, width=8).place(x=30, y=120)
        Button(self.ven, text="Selección", command=self.ordenar_seleccion, width=8).place(x=170, y=120)

        # Etiqueta que muestra el número de elementos
        self.label = Label(self.ven, text="Numeros en la lista: 0")
        self.label.place(x=20, y=160)

        # Listbox para mostrar los datos ingresados
        self.lista = Listbox(self.ven, height=8, width=10, bg="white", fg="black", font=("Helvetica", 12))
        self.lista.place(x=220, y=40)

        # Inicia el bucle principal
        self.ven.mainloop()

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

Programa 3p3

```
from tkinter import *
from tkinter import messagebox
from validar2p3 import validar2
import numpy as np

class Principal:
    def __init__(self):
        self.val = validar2()
        self.ven = Tk()
        self.ven.title('Pactica 3')
        ancho = 350
        alto = 250
        pantalla_ancho = self.ven.winfo_screenwidth()
        pantalla_alto = self.ven.winfo_screenheight()
        x = (pantalla_alto//2)- (ancho//2)
        y = (pantalla_ancho//2)- (alto//2)
        self.ven.geometry(f"{ancho}x{alto}+{x-10}+{y-400}")


    def quitar_placeholder1(self, event):
        if self.nombre.get() == self.placeholder1:
            self.nombre.delete(0, END)
            self.nombre.config(fg="black")

    def quitar_placeholder2(self, event):
        if self.telefono.get() == self.placeholder2:
            self.telefono.delete(0, END)
            self.telefono.config(fg="black")

    def quitar_placeholder3(self, event):
        if self.domicilio.get() == self.placeholder3:
            self.domicilio.delete(0, END)
            self.domicilio.config(fg="black")

    def poner_placeholder1(self, event):
        if self.nombre.get() == "":
```

```
            self.nombre.insert(0, self.placeholder1)
            self.nombre.config(fg="gray")

    def poner_placeholder2(self, event):
        if self.telefono.get() == "":
            self.telefono.insert(0, self.placeholder2)
            self.telefono.config(fg="gray")

    def poner_placeholder3(self, event):
        if self.domicilio.get() == "":
            self.domicilio.insert(0, self.placeholder3)
            self.domicilio.config(fg="gray")

    def inicio(self):
        #caja de texto Nombre
        self.placeholder = "Nombre"
        self.nombre = Entry(self.ven, fg="gray")
        self.nombre.insert(0, self.placeholder)
        self.nombre.bind("<FocusIn>", self.quitar_placeholder1)
        self.nombre.bind("<FocusOut>", self.poner_placeholder1)
        #self.nombre.bind("<Return>", self.validarCaja)
        self.nombre.place(x=50, y=10, width=100)
        #caja de texto Telefono
        self.placeholder = "Telefono"
        self.telefono = Entry(self.ven, fg="gray")
        self.telefono.insert(0, self.placeholder)
        self.telefono.bind("<FocusIn>", self.quitar_placeholder2)
        self.telefono.bind("<FocusOut>", self.poner_placeholder2)
        #self.telefono.bind("<Return>", self.validarCaja)
        self.telefono.place(x=120, y=30, width=100)
        #caja de texto domicilio
        self.placeholder = "Domicilio"
        self.domicilio = Entry(self.ven, fg="gray")
        self.domicilio.insert(0, self.placeholder)
```

```
        self.domicilio.bind("<FocusIn>", self.quitar_placeholder3)
        self.domicilio.bind("<FocusOut>", self.poner_placeholder3)
        self.domicilio.bind("<Return>", self.validarCaja)
        self.domicilio.place(x=230, y=10, width=100)

        Label(self.ven, text="Sexo").place(x=10, y=30)
        self.modo = StringVar(value="F")
        Radiobutton(self.ven, text="F", variable=self.modo, value="F").place(x=10, y=50)
        Radiobutton(self.ven, text="M", variable=self.modo, value="M").place(x=10, y=70)
        self.lista = Listbox(self.ven, height=7, width=40, bg="white", fg="black", font=("Helvetica", 12))
        self.lista.place(x=20, y=100)
        Button(self.ven, text="Agregar", command=self.validarCaja, width=10).place(x=210, y=100)
        self.ven.mainloop()

    def validarCaja(self, evet):
        if (self.nombre.get() == self.placeholder1 or self.telefono.get() == self.placeholder2 or
            self.domicilio == self.placeholder3 or self.domicilio.get()==""):
            messagebox.showerror("Error", "Faltan datos")
        else:
            nombre = self.no  (variable) telefono: Entry
            telefono = self.telefono.get()
            domicilio = self.domicilio.get()
            if self.modo.get() == "F":
                sexo = "Femenino"
            else:
                sexo = "Masculino"
            clave = nombre[0] + telefono[0] + domicilio[2:]
            persona = nombre + " _ "+telefono+" _ "+domicilio+" _ "+sexo
            self.lista.insert(self.lista.size)

if __name__=='__main__':
    app = Principal()
    app.inicio()
```



Programa 4p3

```python
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from validaciones import Validar
import numpy as np
import random

class Principal():
    def __init__(self):
        self.val = Validar()
        self.ven = Tk()
        self.ven.title('Practica 4')
        #self.ven.geometry("500x300")
        ancho = 500
        alto = 300
        ventana_alto = self.ven.winfo_screenwidth()
        ventana_ancho = self.ven.winfo_screenheight()
        x = (ventana_alto // 2) - (ancho // 2)
        y = (ventana_ancho // 2) - (alto // 2)
        self.ven.geometry(f"{ancho}x{alto}+{x}+{y-100}")
        self.cont = 0
        self.bandera = False
        self.renglon = -1
        self.index = 0

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

```
            self.correo.insert(0,valores[2])
            self.bandera= True


    def agregarElemento(self):
        if(len(self.nombre.get())==0 or len(self.edad.get())== 0 or len(self.correo.get())== 0):
            messagebox.showerror("Error","Faltan datos")
        else:
            nombre = self.nombre.get()
            edad = self.edad.get()
            correo = self.correo.get()
            if self.bandera == False:
                self.cont += 1
                clave = str(self.cont)+str(random.randint(1,100))+self.nombre.get()[0:2].upper()
                self.tabla.insert("","end",values=(clave,nombre,correo,edad))
                self.nombre.delete(0,END)
                self.edad.delete(0,END)
                self.correo.delete(0,END)
            else:
                clave = self.index+self.nombre.get()[0:2].upper()
                print("Modo edicion activado")
                self.tabla.item(self.renglon, values=(clave,nombre,correo,edad))
                self.nombre.delete(0,END)
                self.edad.delete(0,END)
                self.correo.delete(0,END)
                self.bandera = False
                self.renglon= -1
                messagebox.showinfo("Correcto","Datos Actualizados")
```

```

    def eliminar(self):
        renglon = self.tabla.selection()
        if not renglon:
            messagebox.showerror("Error","Elige una fila")
        else:
            self.tabla.delete(renglon)
            messagebox.showinfo("Correcto","Fila eliminada")


    def inicio(self):
        Label(self.ven, text="Nombre").place(x=10,y=10)
        self.nombre = Entry(self.ven, fg="blue")
        self.nombre.place(x=10, y=40, width=100)
        Label(self.ven, text="Edad").place(x=130,y=10)
        self.edad = Entry(self.ven, fg="green")
        self.edad.place(x=125, y=40, width=100)
        Label(self.ven, text="Correo").place(x=250,y=10)
        self.correo = Entry(self.ven, fg="purple")
        self.correo.place(x=240, y=40, width=100)
        Button(self.ven, text="Agregar", command=self.agregarElemento, width=10).place(x=380,y=50, width=100,height=30)
        Button(self.ven, text="Eliminar", command=self.eliminar, width=10).place(x=380,y=90, width=100,height=30)
        Button(self.ven, text="Selecionar", command=self.validarCaja, width=10).place(x=380,y=130, width=100,height=30)
        #dataGrid
        columnas = ("Clave","Nombre","Correo","Edad")
        self.tabla = ttk.Treeview(self.ven, columns= columnas, show="headings")
        self.tabla.place(x=10, y=100, width=350,height=190)
        for col in columnas:
            self.tabla.heading(col,text=col)
            self.tabla.column(col, anchor="center", width=30)
        scrolly = ttk.Scrollbar(self.ven,orient="vertical", command=self.tabla.yview)
```

```
        scrollx = ttk.Scrollbar(self.ven, orient="horizontal", command=self.tabla.xview)
        scrolly.place(x=360,y=90,height=200)
        scrollx.place(x=10,y=280, width=350)

        self.ven.mainloop()

if __name__=='__main__':
    app = Principal()
    app.inicio()
```

Validaciones

```python
class Validar:
    def _init_(self):
        self.con = 0

    def validarNumeros(self, num):
        """Verifica que la cadena contenga solo números."""
        if num == "":
            return False

        if self.con >= len(num):
            self.con = 0
            return True

        if 48 <= ord(num[self.con]) <= 57:  # '0' a '9'
            self.con += 1
            return self.validarNumeros(num)
        else:
            self.con = 0
            return False

    def validarLetra(self, dato):
        """Verifica que el primer carácter sea una letra mayúscula."""
        if dato == "":
            return False

        if 65 <= ord(dato[0]) <= 90:  # 'A' a 'Z'
            return True
        else:
            return False

    def validarEntrada(self, dato):
        """Verifica que la cadena tenga máximo 2 dígitos."""
        if dato == "":
            return False
        return len(dato) <= 2
```

Validaciones 2p3

```python

class Validar:
    def __init__(self):
        self.con = 0  # Contador usado en la validación recursiva de números

    def ValidarNumeros(self, num):
        """Verifica que la cadena contenga solo números (recursivo)."""

        # Si la cadena está vacía → no es válida
        if num == "":
            return False

        # Si ya se recorrió toda la cadena → todos los caracteres son válidos
        if self.con >= len(num):
            self.con = 0     # Reinicia el contador
            return True

        # Verifica el carácter actual mediante su código ASCII
        # ord('0') = 48 y ord('9') = 57
        if 48 <= ord(num[self.con]) <= 57:
            self.con += 1
            # Llamada recursiva para revisar el siguiente carácter
            return self.ValidarNumeros(num)
        else:
            # Si encuentra un carácter no numérico, reinicia y retorna False
            self.con = 0
            return False

    def ValidarLetra(self, dato):
        """Verifica que el primer carácter sea una letra mayúscula."""

        # Si está vacío → no es válido
        if dato == "":
            return False

        # Comprueba si el primer carácter está entre 'A' (65) y 'Z' (90)
        if 65 <= ord(dato[0]) <= 90:
            return True
        else:
            return False

    def ValidarEntrada(self, dato):
        """Verifica que la cadena tenga máximo 2 dígitos."""

        # Si está vacío → no es válido
        if dato == "":
            return False

        # Devuelve True si la longitud es de 1 o 2 dígitos
        return len(dato) <= 2
```