MANUAL DE PROGRAMADOR

Francisco Yahir Hernández Ramírez.

Ing. Informática tercer semestre.

Docente: Osvaldo Rene Rojo Roa

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Innovación, Ciencia y Tecnología

Jalisco
GOBIERNO DEL ESTADO

Contenido

PARCIAL 1: PROGRAMA 1. 5
PARCIAL 1: PROGRAMA 2. 6
PARCIAL 1: PROGRAMA 3. 7
PARCIAL 1 PROGRAMA 4. 8
PARCIAL 1 PROGRAMA 5. 9
UNIDAD 1: PROGRAMA 6 10
UNDAD 1: PROGRAMA 7 11
UNIDAD 1: PROGRAMA 8 12
UNIDAD 1: REPASO 1 EXM 13
UNIDAD 1: REPASO 2 EXM 14
UNIDAD 1: REPASO 3 EXMN 15
UNIDAD 2: PROGRAMA 1 17
..... 17
..... 17
..... 17
UNIDAD 2: PROGRAMA 2 18
UNIDAD 2: PROGRAMA 3 20
UNIDAD 2: PROGRAMA 4 22
UNIDAD 2: PRGRAMA 5 24
UNIDAD 2: PROGRAMA 6 27
UNIDAD 2: PROGRAMA 7 30
UNIDAD 2: TAREA 32
UNIDAD 2: REPASO 1 33
UNIDAD 2: REPASO 2 34

```python
# Creamos una lista vacía para guardar las calificaciones
# Create an empty list to store the grades
lista = []

# Declaramos la variable global num
# Declare the global variable num
global num

# Iniciamos el contador en 0
# Initialize the counter at 0
num = 0

# Si este archivo se ejecuta directamente, iniciamos el programa
# If this file is run directly, start the program
if __name__== '__main__':
    inicio(num)
```

PARCIAL 1: PROGRAMA 1.

```python
# Definimos una función llamada inicio que recibe un número (contador)
# Define a function called inicio that receives a number (counter)
def inicio(num):

    # Pedimos al usuario que escriba una calificación y la convertimos a entero
    # Ask the user to enter a grade and convert it to integer
    a = int(input('escribe una calificacion \n'))

    # Incrementamos el contador en 1
    # Increase the counter by 1
    num += 1

    # Agregamos la calificación a la lista
    # Add the grade to the list
    lista.append(a)
```

```python
# Si ya hay 5 o más calificaciones, imprimimos la lista
# If there are 5 or more grades, print the list
if num >= 5:
    print(lista)
else:
    # Si no hay 5 aún, llamamos la función otra vez (recursión)
    # If there are not 5 yet, call the function again (recursion)
    return inicio(num)
```

```python
# Importa todos los elementos del módulo tkinter
# Imports all elements from the tkinter module
from tkinter import *
# Importa la librería messagebox para mostrar cuadros de diálogo
# Imports messagebox to display dialog boxes
from tkinter import messagebox


# Clase que crea una ventana con validación de usuario y contraseña
# Class that creates a window with username and password validation
class Ventana:

    # Constructor de la clase: crea la ventana principal
    # Class constructor: creates the main window
    def __init__(self):
        self.ven = Tk()                             # Crea la ventana principal / Creates the main window
        self.ven.title('Programa 1 con ventanas')   # Título de la ventana / Window title
        self.ven.geometry('400x200')                # Tamaño de la ventana / Window size
        self.inicio()                               # Llama al método que agrega los elementos / Calls the setup method
```

```python
a = int(input('Escribe un numero: '))  # Ask the user to enter a number | Solicita al usuario que ingrese un número
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 número

# 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 operators
/ división exacta con decimales, // división entera, ** potencia, mod módulo, and/or operadores lógicos
'''

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

PARCIAL 1: PROGRAMA 2.

```python
a = [10] # arreglo / array
b =[] # lista / list

a[0] = 10  # asignamos valor 10 al primer elemento / assign value 10 to the first element
a[1] = 10  # intentamos asignar valor en una posición inexistente / trying to assign value in a non-existent position

b={'hola', 10, 10, 5, False,  'm', {1, 2, 3, 4}}
# conjunto (set) con valores repetidos y un set anidado / set with repeated values and a nested set

# ciclos y condiciones / loops and conditions
if (len(a) > len(b)):
    print ('A es mayor')   # se imprime si la lista "a" es más grande / prints if list "a" is larger
else:
    print('B es mayor')    # se imprime si "b" es más grande o igual / prints if "b" is larger or equal

# ciclos / loops
# for
for i in a:
    print(a)  # imprime toda la lista en cada iteración / prints the whole list on each iteration

# foreach (Python usa "for" para recorrer colecciones) / foreach (Python uses "for" to iterate collections)
```

PARCIAL 1: PROGRAMA 3.

```python
# hacer un programa que lea 10 numeros y los almacene en un arreglo
# programa 3
# make a program that reads 10 numbers and stores them in an array
# program 3

a = [0,0,0,0,0,0,0,0,0,0]  # arreglo inicializado con 10 ceros / array initialized with 10 zeros

for i in range(0,10):
    a[i] = int(input('Escribe un numero \n'))
    # se pide un número y se guarda en la posición i / ask for a number and store it in position i
    # La F antes del mensaje es para formatear el dato y agregar una variable para evitar concatenar
    # The F before the message is used to format the string and add a variable instead of concatenating
    # \n salto de linea / \n newline

for i in a:
    print (i)
    # imprime cada número almacenado en el arreglo / prints each number stored in the array
```

PARCIAL 1 PROGRAMA 4.

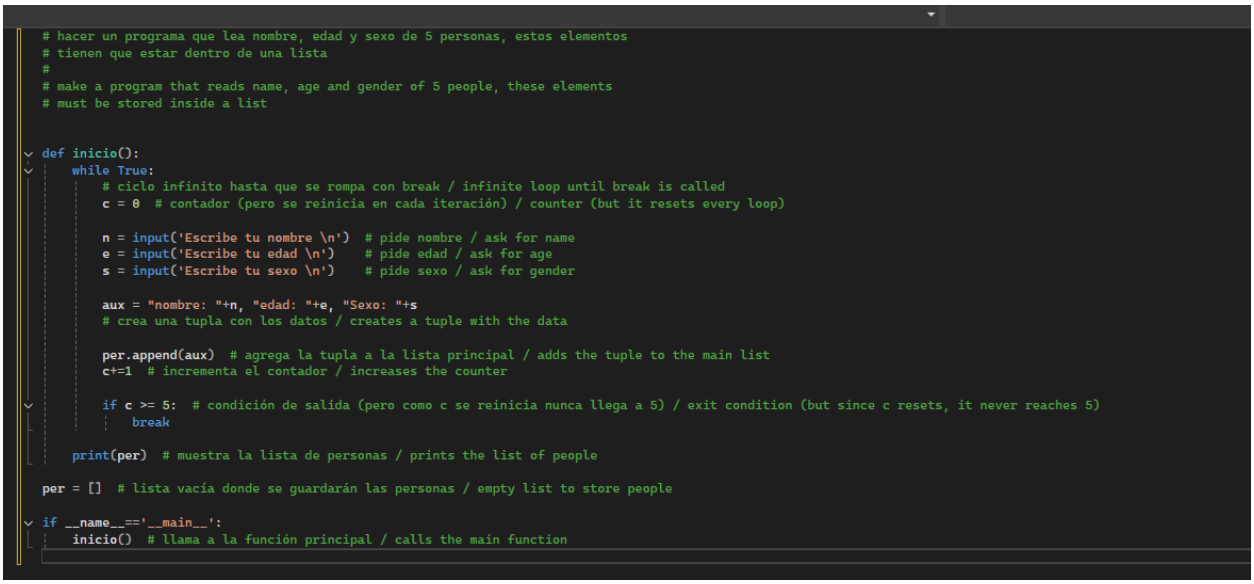
```python
# hacer un  programa que lea 10 numeros y los almacene en una lista
# make a program that reads 10 numbers and stores them in a list

a = []    # lista vacía donde se almacenarán los números / empty list to store the numbers
s = 0     # acumulador para la suma / accumulator for the sum
n = 0     # contador de números ingresados / counter of entered numbers

numeros = "1, 2, 3 ,4 , 5, 6, 7, 8, 9, 0"
# cadena con caracteres permitidos / string with allowed characters

while(n < 10):   # bucle hasta que se ingresen 10 números / loop until 10 numbers are entered
    b =  input('Escribe un numero \n')
    # se pide un número en forma de texto / ask for a number as text
    x = 0   # contador de dígitos válidos / counter of valid digits

    for i in b:   # recorrer cada carácter ingresado / iterate through each entered character
        if i in numeros:
            # si el carácter está en la lista de números permitidos / if the character is in allowed numb
            x += 1

        # otra forma: usando el código ASCII con ord() / another way: using ASCII code with ord()
        # if (ord(i) >= 48 and ord(i) <= 57):

    if len(b) == x:   # si todos los caracteres son números / if all characters are digits
        a.append(int(b))   # convierte a entero y agrega a la lista / convert to int and add to list
        n +=1              # incrementa el contador / increase counter
    else:
        print('El valor no es un numero')   # mensaje de error / error message

for i in a:
    print(i)    # imprime cada número de la lista / print each number from the list
    s+= i       # acumula la suma / accumulate the sum

print(f"La suma es {s}")   # imprime la suma total / print the total sum
```

PARCIAL 1 PROGRAMA 5.

```python
# hacer un programa que lea 10 datos, si el dato es un numero almacenara en un arreglo,
# si es un caracter o caracteres se metera a una lista, cuando finalice el programa nos mostrara
# cuantos elementos nummericos y cuantos caracteres hay en una estructura.
#
# make a program that reads 10 inputs, if the input is a number it will be stored in an array,
# if it is a character or characters it will be stored in a list, when the program ends it will show us
# how many numeric elements and how many characters there are in each structure.


n = [-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]
# arreglo con 11 posiciones inicializado en -1 / array with 11 positions initialized to -1

c = []   # lista vacía para caracteres / empty list for characters
v = 0    # contador de entradas totales / counter for total inputs
v2 = 0   # contador de números válidos (se usa después) / counter for valid numbers (used later)

while(True):  # ciclo infinito controlado por "break" / infinite loop controlled by "break"
    d = input('Escribe un dato o numero \n')
    # pide un dato por teclado / asks for input from keyboard
    v +=1   # incrementa el contador de entradas / increases input counter

    if d.isdigit():
        n[v-1] = int(d)   # si es número, lo guarda en el arreglo en la posición correspondiente / if it's a number, stores it in the array at the position
    elif d.isalpha():
        c.append(d)   # si es letra(s), lo agrega a la lista / if it's alphabetic, adds it to the list

    if v > 10:   # cuando ya se han ingresado más de 10 datos / when more than 10 inputs have been entered
        break
```

```python
    if d.isdigit():
        n[v-1] = int(d)   # si es número, lo guarda en el arreglo en la posición correspondiente / if it's a number, stores it in the array at the position
    elif d.isalpha():
        c.append(d)   # si es letra(s), lo agrega a la lista / if it's alphabetic, adds it to the list

    if v > 10:   # cuando ya se han ingresado más de 10 datos / when more than 10 inputs have been entered
        break

print(f'el arreglo tiene {v2}')  # muestra la cantidad de números (aunque v2 aún no está bien calculado) / shows the count of numbers (though v2 isn't updated yet)
print(f'La lista tiene {len(c)}')  # muestra cuántos caracteres se guardaron / shows how many characters were stored

for i in c:
        if i != -1:   # aquí revisa elementos pero en la lista de caracteres, no en el arreglo / checks elements, but it's iterating over characters list not the array
            v2 += 1   # incrementa el contador / increases the counter

print(c)  # imprime la lista de caracteres / prints the character list
print(n)  # imprime el arreglo de números / prints the number array
```

UNIDAD 1: PROGRAMA 6



```python
# hacer un programa que lea nombre, edad y sexo de 5 personas, estos elementos
# tienen que estar dentro de una lista
#
# make a program that reads name, age and gender of 5 people, these elements
# must be stored inside a list


def inicio():
    while True:
        # ciclo infinito hasta que se rompa con break / infinite loop until break is called
        c = 0  # contador (pero se reinicia en cada iteración) / counter (but it resets every loop)

        n = input('Escribe tu nombre \n')  # pide nombre / ask for name
        e = input('Escribe tu edad \n')    # pide edad / ask for age
        s = input('Escribe tu sexo \n')    # pide sexo / ask for gender

        aux = "nombre: "+n, "edad: "+e, "Sexo: "+s
        # crea una tupla con los datos / creates a tuple with the data

        per.append(aux)  # agrega la tupla a la lista principal / adds the tuple to the main list
        c+=1  # incrementa el contador / increases the counter

        if c >= 5:  # condición de salida (pero como c se reinicia nunca llega a 5) / exit condition (but since c resets, it never reaches 5)
            break

    print(per)  # muestra la lista de personas / prints the list of people

per = []  # lista vacía donde se guardarán las personas / empty list to store people

if __name__=='__main__':
    inicio()  # llama a la función principal / calls the main function
```

UNDAD 1: PROGRAMA 7

```python
# make a program that reads a string and shows how many numbers it has,
# how many uppercase letters, lowercase letters, and spaces


def inicio():

    numeros = "0123456789"
    # cadena con todos los dígitos posibles / string with all possible digits

    cn = 0    # contador de números / counter for numbers
    cmay = 0  # contador de mayúsculas / counter for uppercase letters
    cmin = 0  # contador de minúsculas / counter for lowercase letters
    ce = 0    # contador de espacios / counter for spaces

    cadena = input('Escribe una cadena: \n')
    # pide al usuario una cadena de texto / asks the user for a string
```

```python
    for i in cadena:
        # recorre cada carácter de la cadena / iterates through each character in the string
        if i in numeros:
            cn += 1   # si es número, aumenta contador / if it's a number, increase counter
        if i == ' ':
            ce += 1   # si es espacio, aumenta contador / if it's a space, increase counter
        if ord(i)>=97 and ord(i)<=122:
            cmin +=1  # si es minúscula (código ASCII entre 97 y 122) / if lowercase (ASCII code between 97 and 122)
        if ord(i)>=65 and ord(i)<=90:
                cmay += 1  # si es mayúscula (código ASCII entre 65 y 90) / if uppercase (ASCII code between 65 and 90)

    # muestra resultados / show results
    print(f'Los espacios son: {ce}\n Los numeros: {cn}\n Las mayusculas: {cmay}\n Y las minusculas: {cmin}')

if __name__=='__main__':
        inicio()  # ejecuta la función principal / executes the main function
```

UNIDAD 1: PROGRAMA 8

```python
# hacer un programa que en una lista se introduzca cadenas de caracteres con las siguientes restricciones:
# 1- Las cadenas no deben tener espacios.
# 2- La cadena solo puede tener mayuscula la primer letra.
# 3- Obligatoriamente debe de tener todas las vocales.
# El programa no termina hasta que la lista tenga 5 elementos.
#
# make a program that stores strings in a list with the following restrictions:
# 1- The strings must not have spaces.
# 2- The string can only have the first letter uppercase.
# 3- It must contain all vowels.
# The program will not end until the list has 5 elements.


def inicio():
    a = "abcdefghijklmnopqrstuvwxyz"
    # todas las letras minúsculas permitidas / all allowed lowercase letters
    co = 0   # contador (pero se usa de manera incorrecta para la condición final) / counter (but incorrectly used for final condition)
    l =[]    # lista donde se guardarán las cadenas / list where the strings will be stored

    while(True):  # bucle infinito hasta cumplir condición / infinite loop until condition is met
        c = input('Escribe una cadena: \n')
        # pide al usuario una cadena / asks user for a string

        for i in c[1:]:
            # revisa cada carácter a partir del segundo / checks each character from the second onward
            if i in a:
                co +=1          # si es minúscula, incrementa contador / if lowercase, increase counter
                l.append(c)     # agrega la cadena a la lista (pero sin revisar todas las restricciones) / adds string to the list (but doesn't check all restric
            else:
                print('no hay minusculas')  # mensaje si hay error / message if invalid char
```

```python
    if co <= 5:   # condición de parada (pero mal implementada, se rompe antes de 5 cadenas) / stop condition (but incorrectly implemented, breaks before reaching 5)
        print(l)
        break


if __name__=='__main__':
    inicio()  # ejecuta la función principal / runs the main function
```

UNIDAD 1: REPASO 1 EXM

```python
# instrucciones de entrada y salida
# input and output instructions

# print() o print(f)
# print() or print(f)

#prnt('hola mundo')  # ejemplo con error de sintaxis / example with syntax error
#print(f'hols mundo numeros {10}')  # impresión con formato / formatted print

# entrada datos / data input
#input('escribe un numero') # se introducen solo letras / only letters are typed

# casting para convertir a valores especificos / casting to convert to specific values
#f = float(input('escribe un numero con decimales'))
#a = 0
#a = int(input('escribe un numero'))
#c= 120


# hacer un programa que lea un nombre y precio de un producto, el programa calculara el costo y precio de venta.
# El costo involucra el 12% y el IVA el 16%.
#
# make a program that reads the name and price of a product, the program will calculate the cost and selling price.
# The cost involves 12% and VAT 16%.


# for i in range(1,5) rango de valor inicial hasta el numero final del rango elegido sin incluirlo
# for i in range(1,5) range from initial value to final number without including it
```

```python
while(True):  # ciclo infinito / infinite loop
    n = input('Escribe el nombre del producto: \n')  # pide el nombre del producto / ask for product name
    p = 0.0
    p = float(input('Escribe e [recio del producto: \n'))
    # pide el precio y lo convierte a float / ask for the price and convert to float

    costo = p * 1.12
    # costo incrementado en 12% / cost increased by 12%

    IVA = costo * 1.16
    # costo con 16% adicional de IVA / cost with additional 16% VAT

    (print(f'Producto: {n} \nCosto de venta: {costo:.2f} \nEl precio total es: {IVA:.2f}'))
    # imprime el producto, costo y precio total con 2 decimales / print product, cost and total price with 2 decimals

    res = input('Deseas consultar otro producto? Y/N: \n')
    # pregunta si se desea continuar / ask if user wants to continue

    if(res =='n' or res=='N'):
        break  # si la respuesta es N/n, termina el ciclo / if answer is N/n, loop ends
```

UNIDAD 1: REPASO 2 EXM

```python
a = 1   # coeficiente a de la ecuación cuadrática / coefficient a of the quadratic equation
b = 2   # coeficiente b de la ecuación cuadrática / coefficient b of the quadratic equation
c = -15 # coeficiente c de la ecuación cuadrática / coefficient c of the quadratic equation
p = 0   # variable auxiliar / auxiliary variable
m = 0   # variable auxiliar / auxiliary variable
r = 0   # discriminante / discriminant
ra = 0.0  # raíz cuadrada del discriminante / square root of the discriminant
d = 0.0   # denominador (2a) / denominator (2a)
x1 = 0.0  # primera solución / first solution
x2 = 0.0  # segunda solución / second solution

p = b**2       # calcula b² / calculates b²
m = 4 * a * c  # calcula 4ac / calculates 4ac
r = p - m      # discriminante (b² - 4ac) / discriminant (b² - 4ac)

if r > 0:  # si el discriminante es positivo / if discriminant is positive
    print('Si se puede')  # existen dos soluciones reales / two real solutions exist
    ra = r **(1/2)  # raíz cuadrada del discriminante / square root of discriminant
    d = 2 * a       # denominador = 2a / denominator = 2a
    x1 = (-b + ra)/d  # primera raíz / first root
    x2 = (-b - ra)/d  # segunda raíz / second root
    print(f'El  valor de x1 es: {x1:.2f}\nEl valor de x2 es: {x2:.2f}')
    # imprime las soluciones con 2 decimales / prints the solutions with 2 decimals
else:
    print('Error')  # no existen soluciones reales (discriminante ≤ 0) / no real solutions (discriminant ≤ 0)
```

UNIDAD 1: REPASO 3 EXMN

```python
def validar(a):
    c = 0      # variable auxiliar entera / integer auxiliary variable
    d = 0.0    # variable auxiliar flotante / float auxiliary variable
    try:
        c = int(a)  # intenta convertir a entero / tries to convert to integer
        print('Es un valor numerico sin decimales. ')
        # mensaje si la conversión a entero fue exitosa / message if conversion to integer was successful
    except ValueError:
        print("No es un valor numerico sin decimales. ")
        # mensaje si la conversión a entero falla / message if conversion to integer fails

    try:
        d = float(a)  # intenta convertir a flotante / tries to convert to float
        print('Es un valor numerico con decimales. ')
        # mensaje si la conversión a float fue exitosa / message if conversion to float was successful
    except ValueError:
        print("No es un valor numerico con decimales. ")
        # mensaje si la conversión a float falla / message if conversion to float fails



def leer():
    # ord obtiene el código ASCII del carácter / ord gets the ASCII code of a character
    # isalpha devuelve True si son solo letras / isalpha returns True if only letters
    # isdigit devuelve True si son solo números / isdigit returns True if only numbers
    # try se usa para validar errores de conversión / try is used to handle conversion errors

    a = input('Escribe un dato o valor: \n')  # pide un dato al usuario / asks user for a value
    validar(a)  # llama a la función para validarlo / calls the function to validate it



if __name__=='__main__':
    leer()  # ejecuta la función principal / runs the main function
```

```python
# hacer un programa que lea un dato y que lo almacene en una lista respetando su tipo de dato
# make a program that reads a value and stores it in a list while keeping its data type


def validar(c):
    d = 0.0  # variable auxiliar flotante / float auxiliary variable
    e = 0    # variable auxiliar entera / integer auxiliary variable
    try:
        e = int(c)      # intenta convertir a entero / tries to convert to integer
        return e        # si es entero, lo regresa / if it's integer, return it
    except ValueError:
        print('No es un entero')  # si falla la conversión a entero / if integer conversion fails

    try:
        d = float(c)    # intenta convertir a flotante / tries to convert to float
        return d        # si es flotante, lo regresa / if it's float, return it
    except ValueError:
        print('No es un decimal')  # si falla la conversión a flotante / if float conversion fails

    return c  # si no es número, regresa el valor como cadena / if not a number, return as string
```

```python
def leer():
    c = input('escribe un dato: \n')  # pide un dato al usuario / asks user for a value
    dato = validar(c)                 # valida el tipo de dato / validates the data type
    lista.append(dato)                 # lo agrega a la lista respetando su tipo / adds it to the list keeping its type


lista = []  # lista donde se almacenarán los datos / list where the values will be stored

if __name__=='__main__':
    while(True):  # ciclo infinito hasta que el usuario diga "n" / infinite loop until user types "n"
        leer()
        res = input('Deseas otro? s/n')  # pregunta si se desea ingresar más / ask if user wants to enter more
        if res =='n' or res == 'N':
            print(lista)  # muestra la lista con los datos ingresados / print the list with entered values
            break
```

UNIDAD 2: PROGRAMA 1

```python
# Definimos una función llamada inicio que recibe un número (contador)
# Define a function called inicio that receives a number (counter)
def inicio(num):

    # Pedimos al usuario que escriba una calificación y la convertimos a entero
    # Ask the user to enter a grade and convert it to integer
    a = int(input('escribe una calificacion \n'))

    # Incrementamos el contador en 1
    # Increase the counter by 1
    num += 1

    # Agregamos la calificación a la lista
    # Add the grade to the list
    lista.append(a)
```

```python
 # Si ya hay 5 o más calificaciones, imprimimos la lista
 # If there are 5 or more grades, print the list
 if num >= 5:
     print(lista)
 else:
     # Si no hay 5 aún, llamamos la función otra vez (recursión)
     # If there are not 5 yet, call the function again (recursion)
     return inicio(num)
```

```python
    # Creamos una lista vacía para guardar las calificaciones
    # Create an empty list to store the grades
    lista = []

    # Declaramos la variable global num
    # Declare the global variable num
    global num

    # Iniciamos el contador en 0
    # Initialize the counter at 0
    num = 0

    # Si este archivo se ejecuta directamente, iniciamos el programa
    # If this file is run directly, start the program
    if __name__== '__main__':
        inicio(num)
```

UNIDAD 2: PROGRAMA 2

```python
# Importa la clase 'validacion' desde el archivo validaciones.py
# Imports the 'validacion' class from the file validaciones.py
from validaciones import validacion

# Crea una instancia de la clase validacion y la guarda en la variable 'val'
# Creates an instance of the 'validacion' class and stores it in 'val'
val = validacion()


# Define la clase principal que gestionará las calificaciones
# Defines the main class that will handle the grades
class principal():
    # Método constructor: se ejecuta al crear un objeto de la clase
    # Constructor method: runs when an object of the class is created
    def __init__(self):
        # Crea una lista vacía para almacenar calificaciones
        # Creates an empty list to store grades
        self.lista = []
        # Inicializa un contador de calificaciones
        # Initializes a counter for the number of grades
        self.num = 0
        # Variable para almacenar temporalmente la calificación ingresada
        # Variable to temporarily store the entered grade
        self.a = ""
```

```python
# Método principal que pide al usuario las calificaciones
# Main method that asks the user for grades
def inicio(self):
    # Solicita una calificación al usuario
    # Asks the user to input a grade
    self.a = input('Escribe una calificación \n')

    # Verifica si la entrada es un número válido usando el método de validación
    # Checks if the input is a valid number using the validation method
    if val.validarnumeros(self.a):
        # Incrementa el contador de calificaciones
        # Increments the grade counter
        self.num += 1

        # Convierte la entrada a entero y la agrega a la lista
        # Converts the input to integer and adds it to the list
        self.lista.append(int(self.a))

        # Si ya se ingresaron 5 calificaciones, las muestra en pantalla
        # If 5 grades have been entered, prints them
        if self.num >= 5:
            print(self.lista)
        # Si no, vuelve a pedir otra calificación
        # Otherwise, asks for another grade
```

```
            else:
                self.inicio()
        else:
            # Muestra un mensaje si la entrada no es numérica
            # Displays a message if the input is not a number
            print('No es un número.')
            # Llama nuevamente al método para volver a intentar
            # Calls the method again to retry
            self.inicio()


# Verifica que el script se ejecute directamente (no importado)
# Checks that the script is being run directly (not imported)
if __name__ == '__main__':
    # Crea una instancia de la clase principal
    # Creates an instance of the main class
    app = principal()

    # Llama al método inicio para comenzar el programa
    # Calls the 'inicio' method to start the program
    app.inicio()
```

UNIDAD 2: PROGRAMA 3

```python
# Importa todos los elementos del módulo tkinter
# Imports all elements from the tkinter module
from tkinter import *
# Importa la librería messagebox para mostrar cuadros de diálogo
# Imports the messagebox library to display dialog boxes
from tkinter import messagebox


# Función principal que crea la ventana
# Main function that creates the window
def ventana():

    # Función interna que revisa el usuario y contraseña ingresados
    # Inner function that checks the entered username and password
    def revisar():
        try:
            # Obtiene el texto ingresado en el campo "Usuario"
            # Gets the text entered in the "Usuario" field
            u = str(us.get())
            # Obtiene el texto ingresado en el campo "Password"
            # Gets the text entered in the "Password" field
            p = str(pas.get())
```

```python
            # Verifica si el usuario y la contraseña son correctos
            # Checks if username and password are correct
            if u == 'admin' and p == '1234':
                # Muestra un mensaje de éxito
                # Displays a success message
                messagebox.showinfo('Validación', 'Usuario y contraseña correctos\n(User and password correct)')
            else:
                # Muestra un mensaje de error si los datos no coinciden
                # Displays an error message if credentials are incorrect
                messagebox.showinfo('Error', 'Usuario y/o contraseña incorrectos\n(User and/or password incorrect)'

        # Captura un error si ocurre al procesar los datos
        # Catches an error if something goes wrong during input processing
        except ValueError:
            messagebox.showerror('Error', 'Introduce datos válidos\n(Please enter valid data)')
```

```python
# Crea la ventana principal
# Creates the main window
ven = Tk()
ven.title('Programa 1 con ventanas')  # Título de la ventana / Window ti
ven.geometry('400x200')               # Tamaño de la ventana / Window si

# Etiqueta y campo de entrada para el usuario
# Label and entry for username
Label(ven, text='Usuario:').pack(pady=10)
us = Entry(ven)
us.pack(pady=3)

# Etiqueta y campo de entrada para la contraseña
# Label and entry for password
Label(ven, text='Contraseña:').pack(pady=10)
pas = Entry(ven, show='*')  # Se ocultan los caracteres del password
pas.pack(pady=3)

# Botón para ejecutar la validación
# Button to execute the validation
Button(ven, text='Aceptar', command=revisar).pack(pady=10)

# Inicia el bucle principal de la ventana
# Starts the main loop of the window
ven.mainloop()
```

```python
    # Inicia el bucle principal de la ventana
    # Starts the main loop of the window
    ven.mainloop()


# Verifica que el script se ejecute directamente (no importado)
# Checks that the script is being run directly (not imported)
if __name__ == '__main__':
    ventana()
```

UNIDAD 2: PROGRAMA 4

```
# Importa todos los elementos del módulo tkinter
# Imports all elements from the tkinter module
from tkinter import *
# Importa la librería messagebox para mostrar cuadros de diálogo
# Imports messagebox to display dialog boxes
from tkinter import messagebox


# Clase que crea una ventana con validación de usuario y contraseña
# Class that creates a window with username and password validation
class Ventana:

    # Constructor de la clase: crea la ventana principal
    # Class constructor: creates the main window
    def __init__(self):
        self.ven = Tk()                             # Crea la ventana principal / Creates the main window
        self.ven.title('Programa 1 con ventanas')   # Título de la ventana / Window title
        self.ven.geometry('400x200')                # Tamaño de la ventana / Window size
        self.inicio()                               # Llama al método que agrega los elementos / Calls the setup method
```

```
# Método que construye los elementos gráficos (entradas, etiquetas, botones)
# Method that builds the graphical elements (entries, labels, buttons)
def inicio(self):
    Label(self.ven, text='Usuario:').pack(pady=10)   # Etiqueta para el nombre de usuario / Label for username
    self.us = Entry(self.ven)                        # Campo de entrada para el usuario / Input field for username
    self.us.pack(pady=3)

    Label(self.ven, text='Contraseña:').pack(pady=10)  # Etiqueta para la contraseña / Label for password
    self.pas = Entry(self.ven, show='*')               # Campo de contraseña (oculta el texto) / Password field (hides text)
    self.pas.pack(pady=3)

    # Botón que llama al método revisar cuando se presiona
    # Button that calls the 'revisar' method when clicked
    Button(self.ven, text='Aceptar', command=self.revisar).pack(pady=10)

    # Inicia el bucle principal de la interfaz gráfica
    # Starts the main loop of the GUI
    self.ven.mainloop()
```

```
# Método que valida los datos ingresados
# Method that validates entered data
def revisar(self):
    try:
        # Obtiene los valores de los campos de texto
        # Gets the values from the text fields
        u = str(self.us.get())
        p = str(self.pas.get())

        # Verifica si el usuario y la contraseña son correctos
        # Checks if username and password are correct
        if u == 'admin' and p == '1234':
            messagebox.showinfo('Validación', 'Usuario y contraseña correctos\n(User and password correct)')
        else:
            messagebox.showwarning('Error', 'Usuario y/o contraseña incorrectos\n(User and/or password incorrect)')

    # Captura errores de tipo o valor
    # Catches value or type errors
    except ValueError:
        messagebox.showerror('Error', 'Introduce datos válidos\n(Please enter valid data)')
```

```
            # Captura errores de tipo o valor
            # Catches value or type errors
            except ValueError:
                messagebox.showerror('Error', 'Introduce datos válidos\n(Please enter valid data)')


    # Punto de entrada del programa: crea la ventana al ejecutar el script directamente
    # Entry point of the program: creates the window when the script is run directly
    if __name__ == '__main__':
        app = Ventana()
```

UNIDAD 2: PRGRAMA 5

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

# Clase principal de la aplicación
# Main application class
class principal():
    def _init_(self):
        # Crear ventana principal
        # Create main window
        self.ven = Tk()
        self.ven.title('Programa 5 con ventana')
        self.ven.geometry('600x300')

        # Lista para guardar los promedios individuales
        # List to store individual averages
        self.lista = []

        # Llamamos al método que arma la interfaz
        # Call method that builds the UI
        self.inicio()
```

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

 # Función para calcular el promedio de 4 números ingresados
 # Function to calculate the average of 4 input numbers
 def promediar(self):
     try:
         # Obtener valores de las cajas de texto
         # Get values from entry boxes
         a = float(self.n1.get())
         b = float(self.n2.get())
         c = float(self.n3.get())
         d = float(self.n4.get())

         # Calcular promedio individual
         # Calculate individual average
         pro = (a + b + c + d) / 4

         # Mostrar promedio individual
         # Show individual average
         self.l6.config(text=str(pro))
```

```
    # Guardar promedio en la lista
    # Store average in the list
    self.lista.append(pro)

    # Mostrar lista de promedios
    # Show list of averages
    self.l7.config(text=str(self.lista))

    # Limpiar entradas
    # Clear entry boxes
    self.n1.delete(0, END)
    self.n2.delete(0, END)
    self.n3.delete(0, END)
    self.n4.delete(0, END)

    # Calcular promedio general si hay elementos
    # Calculate general average if list has elements
    if len(self.lista) > 0:
        suma = self.sumar()
        p = suma / len(self.lista)
        self.l8.config(text=f'Promedio general: {str(p)}')

except ValueError:
    # Error si algún dato no es número
    # Error if any value is not a number
```

```
    except ValueError:
        # Error si algún dato no es número
        # Error if any value is not a number
        messagebox.showerror("Error", "Algún dato no es número")
        self.n1.delete(0, END)
        self.n2.delete(0, END)
        self.n3.delete(0, END)
        self.n4.delete(0, END)

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

# Función que crea y posiciona todos los widgets en la ventana
# Function that creates and places all widgets in the window
def inicio(self):
    # Etiquetas y entradas (Labels and entries)
    Label(self.ven, text="Escribe un número").place(y=10, x=20)
    Label(self.ven, text="Escribe un número").place(y=50, x=20)
    Label(self.ven, text="Escribe un número").place(y=90, x=20)
    Label(self.ven, text="Escribe un número").place(y=130, x=20)

    self.n1 = Entry(self.ven)
    self.n1.place(y=10, x=130)
```

```
self.n2 = Entry(self.ven)
self.n2.place(y=50, x=130)

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 para promedio individual (individual avg)
Label(self.ven, text="Promedio").place(y=160, x=130)
self.l6 = Label(self.ven, text="0.0")
self.l6.place(y=160, x=200)

# Botón para calcular el promedio individual
# Button to calculate the individual average
Button(self.ven, text="Promedio", command=self.promediar).place(y=50, x=300)

# Botón para salir
# Button to exit
Button(self.ven, text="Salir", command=self.salir).place(y=90, x=300)

# Etiqueta para mostrar lista de promedios
# Label to display list of averages
self.l7 = Label(self.ven, text="[]")
self.l7.place(y=190, x=200)
```

```
        Button(self.ven, text="Salir", command=self.salir).place(y=90, x=300)

        # Etiqueta para mostrar lista de promedios
        # Label to display list of averages
        self.l7 = Label(self.ven, text="[]")
        self.l7.place(y=190, x=200)

        # Etiqueta para promedio general
        # Label for general average
        self.l8 = Label(self.ven, text="Promedio general: 0.0")
        self.l8.place(y=220, x=150)

        # Iniciar la ventana
        # Start the window loop
        self.ven.mainloop()


# Punto de entrada del programa
# Program entry point
if __name__ == '_main_':
    app = principal()
```

UNIDAD 2: PROGRAMA 6

```python
# Importa todos los componentes del módulo tkinter
# Imports all components from the tkinter module
from tkinter import *
# Importa messagebox para mostrar mensajes emergentes
# Imports messagebox to show pop-up messages
from tkinter import messagebox


# Clase principal que gestiona la interfaz y la lógica del programa
# Main class that manages the interface and program logic
class Principal:
    # Constructor: configura la ventana principal
    # Constructor: sets up the main window
    def __init__(self):
        self.ven = Tk()                                 # Crea la ventana / Creates the window
        self.ven.title('Programa 9 con ventanas')       # Título de la ventana / Window title
        self.ven.geometry('600x300')                    # Tamaño de la ventana / Window size

        # Variables principales
        # Main variables
        self.lista = []     # Lista para almacenar los números / List to store numbers
        self.aux1 = 0       # Variable auxiliar (no usada actualmente) / Auxiliary variable (not used)
        self.aux2 = 0       # Variable auxiliar para inicializar menor / Auxiliary variable for min value
        self.cont = 0       # Contador (no usado en esta versión) / Counter (not used in this version)

        # Llama al método que construye la interfaz
        # Calls the method that builds the interface
```

```python
        # Llama al método que construye la interfaz
        # Calls the method that builds the interface
        self.inicio()

    # Método que crea los elementos de la interfaz gráfica
    # Method that creates the GUI elements
    def inicio(self):
        # Etiquetas y campos de entrada / Labels and input fields
        Label(self.ven, text="Programa 9").grid(row=1, column=2, pady=10)
        Label(self.ven, text="Escribe un número:").grid(row=3, column=1, padx=15, pady=10)
        self.n1 = Entry(self.ven)
        self.n1.grid(row=3, column=2)

        Label(self.ven, text="Escribe otro número:").grid(row=5, column=1, padx=15, pady=10)
        self.n2 = Entry(self.ven)
        self.n2.grid(row=5, column=2)

        # Botones de acción / Action buttons
        Button(self.ven, text="AGREGAR", command=self.agregar).grid(row=6, column=1, padx=10)
        Button(self.ven, text="MAYOR", command=self.mayor).grid(row=6, column=2, padx=10)
        Button(self.ven, text="MENOR", command=self.menor).grid(row=6, column=3, padx=10)
        Button(self.ven, text="SALIR", command=self.salir).grid(row=6, column=4, padx=10)
```

```
        # Listbox para mostrar los elementos agregados
        # Listbox to display added elements
        self.listview = Listbox(self.ven, height=10, width=15, bg='lightgrey', activestyle="d
        self.listview.grid(row=2, column=4, rowspan=5, padx=20)

        # Etiqueta para mostrar la lista actual
        # Label to display the current list
        self.listaElementos = Label(self.ven, text="")
        self.listaElementos.grid(row=8, column=2, pady=15)

        # Inicia el bucle principal de la ventana
        # Starts the main loop of the window
        self.ven.mainloop()

    # Método para agregar los números a la lista
    # Method to add numbers to the list
    def agregar(self):
        try:
            # Convierte los valores ingresados a enteros
            # Converts entered values to integers
            a = int(self.n1.get())
            b = int(self.n2.get())

            # Agrega ambos números a la lista
            # Adds both numbers to the list
            self.lista.extend([a, b])
```

```
            # Actualiza la etiqueta con la lista actual
            # Updates the label with the current list
            self.listaElementos.config(text=f'Lista actual: {self.lista}')

        except ValueError:
            # Muestra un error si los datos no son válidos
            # Shows an error if input data are invalid
            messagebox.showerror("Error", "Algún dato no es un número / Some input is not a number")

    # Método para mostrar el número mayor
    # Method to display the largest number
    def mayor(self):
        if self.lista:
            mayor_valor = max(self.lista)
            messagebox.showinfo("Resultado", f'El número mayor es: {mayor_valor}\n(The largest number is: {mayor_valor})')
        else:
            messagebox.showerror("Error", "La lista está vacía / The list is empty")

    # Método para mostrar el número menor
    # Method to display the smallest number
    def menor(self):
        if self.lista:
            menor_valor = min(self.lista)
            messagebox.showinfo("Resultado", f'El número menor es: {menor_valor}\n(The smallest number is: {menor_valor})')
        else:
            messagebox.showerror("Error", "La lista está vacía / The list is empty")
```

```
    def menor(self):
        if self.lista:
            menor_valor = min(self.lista)
            messagebox.showinfo("Resultado", f'El número menor es: {menor_valor}\n(The smallest number is: {menor_valor})')
        else:
            messagebox.showerror("Error", "La lista está vacía / The list is empty")

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


# Punto de entrada principal del programa
# Main entry point of the program
if __name__ == '__main__':
    Principal()
```

UNIDAD 2: PROGRAMA 7

```python
# Importa todos los elementos del módulo tkinter
# Imports all elements from the tkinter module
from tkinter import *
# Importa messagebox para mostrar cuadros de diálogo
# Imports messagebox to display dialog boxes
from tkinter import messagebox


# Clase principal que controla la interfaz gráfica y la lógica del programa
# Main class that controls the GUI and program logic
class Principal:
    def __init__(self):
        # Crea la ventana principal / Creates the main window
        self.ven = Tk()
        self.ven.title('Programa 10 con ventanas')  # Título de la ventana / Window title
        self.ven.geometry('600x300')                # Tamaño de la ventana / Window size

        # Variables para almacenar datos / Variables to store data
        self.lista = []     # Lista de números / List of numbers
        self.aux1 = 0       # Variables auxiliares (no se usan actualmente) / Auxiliary variables (not used)
        self.aux2 = 0
        self.cont = 0

        # Llama al método que construye la interfaz / Calls the method that builds the interface
        self.inicio()
```

```python
# Método que coloca los elementos gráficos en la ventana
# Method that places the graphical elements in the window
def inicio(self):
    # Etiquetas y campos de entrada / Labels and entry fields
    Label(self.ven, text="Programa 10").place(x=10, y=10)
    Label(self.ven, text="Escribe un número:").place(x=10, y=50)
    self.n1 = Entry(self.ven)
    self.n1.place(x=150, y=50)

    Label(self.ven, text="Escribe otro número:").place(x=10, y=90)
    self.n2 = Entry(self.ven)
    self.n2.place(x=150, y=90)

    # Botones con funciones asociadas / Buttons with their respective functions
    Button(self.ven, text="AGREGAR", command=self.agregar).place(x=10, y=130)
    Button(self.ven, text="MAYOR", command=self.mayor).place(x=110, y=130)
    Button(self.ven, text="MENOR", command=self.menor).place(x=210, y=130)
    Button(self.ven, text="SALIR", command=self.salir).place(x=310, y=130)

    # Listbox para mostrar los elementos agregados / Listbox to show added numbers
    self.listview = Listbox(
        self.ven, height=10, width=15, bg='lightgrey', activestyle="dotbox"
    )
    self.listview.place(x=450, y=30)
```

```
    # Etiqueta para mostrar la lista actual / Label to show the current list
    self.listaElementos = Label(self.ven, text="")
    self.listaElementos.place(x=150, y=180)

    # Inicia el ciclo principal de la interfaz / Starts the GUI main loop
    self.ven.mainloop()

# Método para agregar los números a la lista / Adds the entered numbers to the list
def agregar(self):
    try:
        # Convierte los valores de entrada a enteros / Converts input values to integers
        a = int(self.n1.get())
        b = int(sel
                (parameter) self: Self@Principal

        # Agrega ambos números a la lista / Adds both numbers to the list
        self.lista.extend([a, b])

        # Inserta los valores en el Listbox / Inserts values into the Listbox
        self.listview.insert(END, a)
        self.listview.insert(END, b)

        # Limpia los campos de entrada / Clears the entry fields
        self.n1.delete(0, END)
        self.n2.delete(0, END)

        # Actualiza la etiqueta con la lista actual / Updates label with the current list
        self.listaElementos.config(text='Lista actual: ' + str(self.lista))
```

```
    # Método que muestra el número mayor / Shows the largest number
    def mayor(self):
        if self.lista:
            mayor_valor = max(self.lista)
            messagebox.showinfo("Resultado", f'El número mayor es: {mayor_valor}\n(The largest number is: {mayor_valor})')
        else:
            messagebox.showerror("Error", "La lista está vacía / The list is empty")

    # Método que muestra el número menor / Shows the smallest number
    def menor(self):
        if self.lista:
            menor_valor = min(self.lista)
            messagebox.showinfo("Resultado", f'El número menor es: {menor_valor}\n(The smallest number is: {menor_valor})')
        else:
            messagebox.showerror("Error", "La lista está vacía / The list is empty")

    # Método para cerrar la ventana / Closes the window
    def salir(self):
        self.ven.destroy()


# Punto de entrada del programa / Program entry point
if __name__ == '__main__':
    Principal()
```

UNIDAD 2: TAREA

```python
import tkinter as tk                    # EN: Import Tkinter library for GUI | ES: Importar la librería Tkinter para interfaz gráfica
from tkinter import messagebox, simpledialog   # EN: Import dialog boxes and messages | ES: Importar cuadros de diálogo y mensajes

lista2 = []  # EN: This list will store all people with their grades | ES: Esta lista guardará todas las personas con sus calificaciones


def pedir_datos():
    # EN: Ask for the name of the person | ES: Pedir el nombre de la persona
    nom = simpledialog.askstring('Entrada', 'Escribe un nombre:')

    if not nom:  # EN: If no name is entered, exit the function | ES: Si no se escribe nombre, salir de la función
        return

    # EN: Ask for 3 grades and convert them to integers | ES: Pedir 3 calificaciones y convertirlas a enteros
    try:
        c1 = int(simpledialog.askstring('Entrada', 'Escribe una calificacion:'))
        c2 = int(simpledialog.askstring('Entrada', 'Escribe una calificacion:'))
        c3 = int(simpledialog.askstring('Entrada', 'Escribe una calificacion:'))
    except:
        # EN: If the user types something that is not a number, show an error | ES: Si el usuario escribe algo que no es número, mostrar error
        messagebox.showerror('Error', 'Debes escribir numeros enteros')
        return

    # EN: Sort grades from highest to lowest | ES: Ordenar las calificaciones de mayor a menor
    calificaciones = sorted([c1, c2, c3], reverse=True)
```

```python
        # EN: Add the list to the global list of people | ES: Agregar la lista a la lista global de personas
        lista2.append(lista1)

        # EN: Show a message that the record was added | ES: Mostrar un mensaje de que se agregó el registro
        messagebox.showinfo('Registro agregado', f'Se agrego:\n{lista1}')


def mostrar_lista():
    # EN: If the list is empty, show a message | ES: Si la lista está vacía, mostrar mensaje
    if not lista2:
        messagebox.showinfo('Lista', 'No hay datos todavia.')
    else:
        # EN: Convert the list into text to display | ES: Convertir la lista en texto para mostrar
        texto = '\n'.join([str(p) for p in lista2])
        messagebox.showinfo('Lista de Personas', texto)


def salir():
    # EN: Close the program window | ES: Cerrar la ventana del programa
    root.destroy()
```

```python
# ---------- MAIN WINDOW / VENTANA PRINCIPAL ----------
root = tk.Tk()                          # EN: Create the main window | ES: Crear la ventana principal
root.title('Registro de Calificaciones') # EN: Set the window title | ES: Poner título a la ventana
root.geometry('300x200')                # EN: Define window size | ES: Definir el tamaño de la ventana

# ---------- BUTTONS / BOTONES ----------
btn_agregar = tk.Button(root, text='Agregar Persona', command=pedir_datos, width=20)  # EN: Button to add a person | ES: Botón para agregar persona
btn_agregar.pack(pady=10)  # EN: Place button with vertical margin | ES: Colocar el botón con margen vertical

btn_mostrar = tk.Button(root, text='Mostrar Lista', command=mostrar_lista, width=20)  # EN: Button to show all data | ES: Botón para mostrar todos los datos
btn_mostrar.pack(pady=10)

btn_salir = tk.Button(root, text='Salir', command=salir, width=20)  # EN: Button to exit | ES: Botón para salir
btn_salir.pack(pady=10)

# ---------- LOOP / BUCLE PRINCIPAL ----------
root.mainloop()  # EN: Start the window loop to keep program running | ES: Iniciar el bucle de la ventana para mantener el programa abierto
```

UNIDAD 2: REPASO 1

UNIDAD 2: REPASO 2

```python
from tkinter import *             # Import tkinter for GUI / Importa tkinter para la interfaz grafica
from tkinter import messagebox    # Import messagebox for alerts / Importa messagebox para mostrar alertas


class Principal():                # Define main class / Define la clase principal
    def __init__(self):           # Constructor method / Metodo constructor
        self.ven = Tk()           # Create window / Crea la ventana
        self.ven.title('Examen Caliz')    # Set window title / Establece el titulo de la ventana
        self.ven.geometry('600x300')      # Set window size / Define el tamano de la ventana
        self.lista = []           # Create empty list / Crea una lista vacia
        self.Inicio()             # Call start method / Llama al metodo de inicio

    def Inicio(self):             # Define start method / Define el metodo de inicio
        # Labels for textboxes / Etiquetas para las cajas de texto
        Label(self.ven, text="Letras minusculas:").place(x=20, y=20)   # Label for lowercase / Etiqueta para minusculas
        Label(self.ven, text="Letras MAYUSCULAS:").place(x=220, y=20)  # Label for uppercase / Etiqueta para mayusculas
        Label(self.ven, text="Numeros:").place(x=420, y=20)            # Label for numbers / Etiqueta para numeros
```

```python
    def Inicio(self):             # Define start method / Define el metodo de inicio
        # Labels for textboxes / Etiquetas para las cajas de texto
        Label(self.ven, text="Letras minusculas:").place(x=20, y=20)   # Label for lowercase / Etiqueta para minusculas
        Label(self.ven, text="Letras MAYUSCULAS:").place(x=220, y=20)  # Label for uppercase / Etiqueta para mayusculas
        Label(self.ven, text="Numeros:").place(x=420, y=20)            # Label for numbers / Etiqueta para numeros

        # Entry boxes / Cajas de texto
        self.min = Entry(self.ven)                    # Create entry for lowercase / Crea entrada para minusculas
        self.min.place(x=20, y=40, width=150)         # Place it in window / La coloca en la ventana

        self.may = Entry(self.ven)                    # Create entry for uppercase / Crea entrada para mayusculas
        self.may.place(x=220, y=40, width=150)        # Place it in window / La coloca en la ventana

        self.num = Entry(self.ven)                    # Create entry for numbers / Crea entrada para numeros
        self.num.place(x=420, y=40, width=150)        # Place it in window / La coloca en la ventana

        # Buttons / Botones
        Button(self.ven, text="VALIDAR", command=self.validar).place(x=180, y=100)  # Validate button / Boton para validar
        Button(self.ven, text="AGREGAR", command=self.agregar).place(x=320, y=100)  # Add button / Boton para agregar

        # Listbox and counter label / Listbox y etiqueta de contador
        self.lista_visual = Listbox(self.ven, width=70)  # Create listbox / Crea un listbox
        self.lista_visual.place(x=50, y=150)             # Place it in window / Lo coloca en la ventana

        self.contador = Label(self.ven, text="Elementos: 0")  # Label to show number of elements / Etiqueta para mostrar cantidad de ele
        self.contador.place(x=50, y=250)                      # Place it / La coloca en la ventana
```

```python
def validar(self):        # Method to validate inputs / Metodo para validar las entradas
    correcto = True       # Flag variable for validation / Variable de control para validacion

    # Validate lowercase text / Validar texto en minusculas
    texto_min = self.min.get()     # Get lowercase entry / Obtiene texto de minusculas
    if not texto_min.islower() or texto_min == "":   # If not lowercase or empty / Si no es minuscula o esta vacio
        messagebox.showerror("Error", "La primera caja debe contener solo letras minusculas.")  # Show error / Muestra error
        self.min.delete(0, END)     # Clear box / Borra la caja
        correcto = False            # Mark as incorrect / Marca como incorrecto

    # Validate uppercase text / Validar texto en mayusculas
    texto_may = self.may.get()      # Get uppercase entry / Obtiene texto de mayusculas
    if not texto_may.isupper() or texto_may == "":   # If not uppercase or empty / Si no es mayuscula o esta vacio
        messagebox.showerror("Error", "La segunda caja debe contener solo letras MAYUSCULAS.")  # Show error / Muestra error
        self.may.delete(0, END)     # Clear box / Borra la caja
        correcto = False            # Mark as incorrect / Marca como incorrecto

    # Validate number text / Validar texto numerico
    texto_num = self.num.get()      # Get number entry / Obtiene texto de numeros
    if not texto_num.isdigit() or texto_num == "":   # If not number or empty / Si no es numero o esta vacio
        messagebox.showerror("Error", "La tercera caja debe contener solo numeros.")  # Show error / Muestra error
        self.num.delete(0, END)     # Clear box / Borra la caja
        correcto = False            # Mark as incorrect / Marca como incorrecto

    if correcto:                    # If all correct / Si todo es correcto
        messagebox.showinfo("Validacion exitosa", "Todas las entradas son validas.")  # Show success / Muestra mensaje exitoso
```

```
        messagebox.showinfo("Validacion exitosa", "Todas las entradas son validas.")  # Show success / Muestra mensaje exitoso

    def agregar(self):        # Method to add validated entries / Metodo para agregar las entradas validadas
        texto_min = self.min.get()   # Get lowercase / Obtiene minusculas
        texto_may = self.may.get()   # Get uppercase / Obtiene mayusculas
        texto_num = self.num.get()   # Get number / Obtiene numeros

        # Check that fields are not empty / Verifica que no esten vacios
        if texto_min and texto_may and texto_num:
            # Check they are valid types / Verifica que sean tipos correctos
            if texto_min.islower() and texto_may.isupper() and texto_num.isdigit():
                concatenado = texto_min + texto_may + texto_num   # Concatenate all / Concatena todos los textos
                self.lista.append(concatenado)                    # Add to list / Agrega a la lista
                self.lista_visual.insert(END, concatenado)        # Show in listbox / Lo muestra en el listbox
                self.contador.config(text=f"Elementos: {len(self.lista)}")  # Update count / Actualiza el contador

                # Clear all entries / Limpia todas las cajas
                self.min.delete(0, END)
                self.may.delete(0, END)
                self.num.delete(0, END)
            else:
                messagebox.showerror("Error", "Debe validar correctamente las entradas antes de agregar.")  # Show error / Muestra error
        else:
            messagebox.showerror("Error", "No puede haber campos vacios.")  # Show error if empty / Muestra error si hay vacios
```

```
if __name__ == '__main__':   # Main entry point / Punto de entrada principal
    app = Principal()         # Create app instance / Crea una instancia de la aplicacion
```