



## MANUAL DE PROGRAMACION

TECNOLOGICO JOSE MARIO MOLINA PASQUEL Y HENRIQUEZ  
ZAPOTLANEJO



## ESTRUCTURA DE DATOS

LESLIE DAREYA RUIZ BARRERA

Informática 3



## Índice

REPASO - Programa 1 .....	2
PRACTICA 2.....	3
PROGRAMA 3 .....	4
PROGRAMA 4 – Tarea .....	6
PROGRAMA 5 .....	9
PROGRAMA 6 .....	11
PROGRAMA 7 .....	13
PROGRAMA 8 .....	14
PROGRAMA 9 .....	16
REPASO 1.....	18
REPASO 2.....	18
REPASO 3.....	18
PRACTICA REPASO 3.....	18



## REPASO - Programa 1

```

2   a = [10] #arreglo / array
3   b = [] # una lista no tiene tamaño es indefinida / a list has no fixed size, it is undefined
4
5   a[0] = 10
6   a[1] = 10
7
8   b={'hola',10,10.05,False, 'm' ,{1,2,3,4}}
9
10  #operadores logicos / logical operators
11  # + - *
12  # mod
13  # and or
14  # ** potencia o elevar a / power or exponentiation
15  # / exacta, con decimales / exact, with decimals
16  #// sin decimales / without decimals
17  # < > >= <= != not ==
18
19  #ciclos y condiciones / loops and conditions
20  if (len(a) > len(b)):
21      print('A es mayor')  # A es mayor / A is greater
22  else:
23      print('B es mayor')  # B es mayor / B is greater
24

```

### Agregar un numero en cadena y mandarlo a llamar

```

1   print('Escribe un numero')
2   # Muestra un mensaje en pantalla / Displays a message on the screen
3
4   a = input('Escribe un numero')
5   # Pide al usuario que ingrese un valor y lo guarda en la variable 'a'
6   # / Asks the user to enter a value and stores it in the variable 'a'
7
8   print(a)
9   # Imprime en pantalla el valor que se guardó en la variable 'a'
10  # / Prints on the screen the value stored in the variable 'a'

```

```

Escribe un numero
Escribe un numero2
2

```



## PRACTICA 2

### Arreglos, Listas, Ciclos y condiciones

```
1  a = []
2  # Crea una lista vacía llamada a donde se guardarán los números válidos que el usuario introduce.
3  # Creates an empty list called a where the valid numbers entered by the user will be stored.
4  s = 0
5  # Inicia la variable s en 0. Se usará como acumulador para sumar los números guardados.
6  # Initializes the variable s to 0. It will be used as an accumulator to sum the stored numbers.
7  n = 0
8  # Contador n en 0. Cuenta cuántos números válidos se han guardado en a
9  # Counter n set to 0. Counts how many valid numbers have been stored in a
10 numeros = "0123456789"
11 # Cadena con los caracteres permitidos para considerar una entrada como número.
12 # String with the allowed characters to consider an entry as a number.
13 while(n < 10):
14     # Bucle while que se repite mientras n sea menor que 10
15     # While loop that repeats while n is less than 10
16     b = input('Escribe un numero: ')
17     # Pide al usuario que escriba algo y lo guarda como cadena en b
18     # Asks the user to enter something and stores it as a string in b
19     x = 0
20     # x servirá para contar cuántos caracteres de la cadena b son dígitos
21     # x will be used to count how many characters of the string b are digits
22     for i in b:
23         # RECORRE LOS ELEMENTOS B
24         # ITERATES THROUGH THE ELEMENTS OF b
25         if i in numeros:
26             x += 1
27             # Si el carácter i está en la cadena numeros se incrementa x
28             # If the character i is in the string numeros, x is incremented
29     if len(b) == x:
30         # Si son iguales, se considera la entrada válida como número entero.
31         # If they are equal, the entry is considered valid as an integer number.
32         a.append(int(b))
33         # Convierte la cadena b a entero con int(b) y la añade a la lista a.
34         # Converts the string b to an integer with int(b) and adds it to the list a.
35         n += 1
36         # Incrementa n porque se guardó un número válido.
37         # Increments n because a valid number was stored.
38     else:
39         print('El valor no es numero ')
40         # Si la entrada no está compuesta únicamente por dígitos, muestra el mensaje de error
41         # If the entry is not composed only of digits, displays the error message
```



## PROGRAMA 3

**Hacer un programa que lea 10 números y los almacene en un arreglo**

```
1  a = [0,0,0,0,0,0,0,0,0,0]
2  # Crea una lista de 10 elementos inicializados en 0 / Creates a list of 10 elements initialized to 0
3
4  for i in range(0,10):
5      # Repite las instrucciones 10 veces, con 'i' desde 0 hasta 9
6      # / Repeats the instructions 10 times, with 'i' from 0 to 9
7      a[i] = int(input('Escribe un numero \n'))
8      # Pide al usuario un número, lo convierte a entero y lo guarda en la posición i de la lista
9      # / Asks the user for a number, converts it to integer, and stores it at position i in the list
10
11 for i in a:
12     # Recorre cada elemento de la lista / Iterates through each element in the list
13     print(i)
14     # Imprime el valor de cada elemento / Prints the value of each element
```

Escribe un numero  
1  
Escribe un numero  
2  
Escribe un numero  
3  
Escribe un numero  
4  
Escribe un numero  
5  
Escribe un numero  
6  
Escribe un numero  
7  
Escribe un numero  
8  
Escribe un numero  
9  
Escribe un numero  
0

1  
2  
3  
4  
5  
6  
7  
8  
9  
0



## Opción 2 programa 3

```

1  array = []
2  # Crea una lista vacía llamada 'array'
3  # / Creates an empty list called 'array'
4
5  while True:
6      # Inicia un bucle infinito que seguirá ejecutándose hasta que se cumpla un 'break'
7      # / Starts an infinite loop that will continue until a 'break' is reached
8
9      contador = len(array)
10     # Guarda en 'contador' la cantidad de elementos que hay actualmente en la lista
11     # / Stores in 'contador' the number of elements currently in the list
12
13     entrada_usurio = input("Escribe un numero por favor: ")
14     # Pide al usuario que ingrese un valor y lo guarda en la variable 'entrada_usurio'
15     # / Asks the user to enter a value and stores it in 'entrada_usurio'
16
17     if entrada_usurio.isdigit():
18         # Verifica si lo ingresado por el usuario es un número entero positivo (solo dígitos)
19         # / Checks if the user input is a positive integer (digits only)
20         array.append(entrada_usurio)
21         # Si es un número válido, lo agrega al final de la lista
22         # / If valid, appends it to the end of the list
23         print(f"Lista completa: {array}")
24         # Muestra en pantalla cómo se ve la lista después de la posible adición
25         # / Shows on screen how the list looks after the possible addition
26
27     if contador >= 9:
28         # Si la cantidad de elementos antes de esta iteración era 10 o más, termina el bucle
29         # / If the number of elements before this iteration was 10 or more, it breaks the loop
30         break

```

```

Escribe un numero por favor: 1
Lista completa: ['1']
Escribe un numero por favor: 2
Lista completa: ['1', '2']
Escribe un numero por favor: 3
Lista completa: ['1', '2', '3']
Escribe un numero por favor: 4
Lista completa: ['1', '2', '3', '4']
Escribe un numero por favor: 5
Lista completa: ['1', '2', '3', '4', '5']
Escribe un numero por favor: 6
Lista completa: ['1', '2', '3', '4', '5', '6']
Escribe un numero por favor: 7
Lista completa: ['1', '2', '3', '4', '5', '6', '7']
Escribe un numero por favor: 8
Lista completa: ['1', '2', '3', '4', '5', '6', '7', '8']
Escribe un numero por favor: 9
Lista completa: ['1', '2', '3', '4', '5', '6', '7', '8', '9']
Escribe un numero por favor: 0
Lista completa: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0']

```



## PROGRAMA 4 – Tarea

**Hacer un programa que lea 10 números, los almacene en una lista y luego los sume mostrando el resultado final**

```
5  a = []
6  # Crea una lista vacía para guardar los números / Creates an empty list to store the numbers
7
8  s = 0
9  # Inicializa la variable 's' para almacenar la suma / Initializes the variable 's' to store the sum
10
11 n = 0
12 # Inicializa la variable 'n' para contar cuántos números válidos se han ingresado
13 # / initializes 'n' to count how many valid numbers have been entered
14
15 numeros = "0,1,2,3,4,5,6,7,8,9";
16 # Define una cadena con los caracteres numéricos válidos
17 # / Defines a string with the valid numeric characters
18
19 while(n < 10):
20     # Bucle que se repite hasta que se hayan ingresado 10 números válidos
21     # / Loop that repeats until 10 valid numbers have been entered
22
23     b = input('Escribe un numero')
24     # Pide al usuario que ingrese un número y lo guarda en 'b'
25     # / Asks the user to enter a number and stores it in 'b'
26
27     x = 0
28     # Inicializa un contador 'x' para verificar si todos los caracteres son dígitos
29     # / initializes a counter 'x' to check if all characters are digits
30
31     for i in b:
32         # Recorre cada carácter en la entrada 'b'
33         # / Iterates over each character in the input 'b'
```



```

34
35      # if (ord(i) >= 48 and ord(i) <=57):
36  v   if i in numeros:
37      # Verifica si el carácter está en la lista de números válidos
38      # / Checks if the character is in the list of valid numbers
39      x += 1
40      # Si es válido, aumenta el contador 'x'
41      # / If valid, increments the counter 'x'
42
43  v   if len(b) == x:
44      # Si todos los caracteres son dígitos, se considera un número válido
45      # / If all characters are digits, it is considered a valid number
46      a.append(int(b))
47      # Convierte la entrada a entero y la agrega a la lista 'a'
48      # / Converts the input to an integer and appends it to the list 'a'
49      n += 1
50      # Aumenta el contador de números válidos ingresados
51      # / Increments the count of valid numbers entered
52  v   else:
53      print('el valor no es un numero')
54      # Si algún carácter no es dígito, muestra un mensaje de error
55      # / If any character is not a digit, shows an error message
56
57  v for i in a:
58      # Recorre cada número de la lista 'a' / Iterates over each number in the list 'a'
59      print(i)
60      # Imprime el número actual / Prints the current number
61      s += i
62      # Suma el número actual a la variable 's' / Adds the current number to the variable 's'
63
64  print(f'La suma es {s}')
65  # Imprime la suma total de los números ingresados / Prints the total sum of the entered numbers

```

```

Escribe un numero1
Escribe un numero2
Escribe un numero3
Escribe un numero4
Escribe un numero5
Escribe un numero6
Escribe un numero7
Escribe un numero8
Escribe un numero9
Escribe un numero0
1
2
3
4
5
6
7
8
9
0
La suma es 45

```

Opción 2 programa 4



```
3 lista = []
4 # Crea una lista vacía para almacenar los números ingresados
5 # / Creates an empty list to store the entered numbers
6
7 for e in range(0,9):
8     # Bucle que se repite 9 veces / Loop that repeats 9 times
9     numeros = input("Ingresa un numero: ")
10    # Pide al usuario que ingrese un número y lo guarda en 'numeros'
11    # / Asks the user to enter a number and stores it in 'numeros'
12    x = numeros
13    # Guarda el valor de 'numeros' en la variable 'x'
14    # / Stores the value of 'numeros' in the variable 'x'
15    lista.append(int(x))
16    # Convierte 'x' a entero y lo agrega a la lista
17    # / Converts 'x' to integer and appends it to the list
18    print(f"La lista ahora tiene estos elementos: {lista}")
19    # Muestra cómo se ve la lista después de agregar el número
20    # / Shows how the list looks after adding the number
21
22 suma_total = 0
23 # Inicializa la variable para almacenar la suma de los números
24 # / Initializes the variable to store the sum of the numbers
25
26 for e in lista:
27     # Recorre cada elemento de la lista / Iterates over each element in the list
28     suma_total = suma_total + e
29     # Suma cada elemento al total / Adds each element to the total
30
31 print(f"La suma total de todos los numeros es: {suma_total}")
32 # Imprime la suma total de los números ingresados / Prints the total sum of the entered numbers
```

```
Ingresa un numero: 1
La lista ahora tiene estos elementos: [1]
Ingresa un numero: 2
La lista ahora tiene estos elementos: [1, 2]
Ingresa un numero: 3
La lista ahora tiene estos elementos: [1, 2, 3]
Ingresa un numero: 4
La lista ahora tiene estos elementos: [1, 2, 3, 4]
Ingresa un numero: 5
La lista ahora tiene estos elementos: [1, 2, 3, 4, 5]
Ingresa un numero: 6
La lista ahora tiene estos elementos: [1, 2, 3, 4, 5, 6]
Ingresa un numero: 7
La lista ahora tiene estos elementos: [1, 2, 3, 4, 5, 6, 7]
Ingresa un numero: 8
La lista ahora tiene estos elementos: [1, 2, 3, 4, 5, 6, 7, 8]
Ingresa un numero: 9
La lista ahora tiene estos elementos: [1, 2, 3, 4, 5, 6, 7, 8, 9]
La suma total de todos los numeros es: 45
```



## PROGRAMA 5

**Hacer un programa que lea 10 datos, si el dato es un número se almacenará en un arreglo, si es un carácter o caracteres se meterá a una lista, cuando finalice el programa nos mostrará cuántos elementos numéricos y cuántos caracteres hay en cada estructura**

```

6  arr = [-1,-1,-1,-1,-1,-1,-1,-1] # arreglo para almacenar números / array to store numbers
7  car = [] # lista para almacenar caracteres / list to store characters
8  c = 0 # contador para el arreglo / counter for the array
9  c2 = 0 # contador para números en el arreglo / counter for numbers in the array
10
11 while(True):
12     a = input('Escribe un dato o valor\n') # pedir al usuario que ingrese un valor / ask the user to enter a value
13     if a.isdigit(): # verificar si la entrada es un número / check if the input is a digit
14         arr[c] = int(a) # almacenar el número en el arreglo / store the number in the array
15     elif a.isalpha(): # verificar si la entrada es una letra o palabra / check if the input is alphabetic
16         car.append(a) # agregar la entrada a la lista de caracteres / add the input to the character list
17     c += 1 # incrementar el contador / increment the counter
18     if c >= 10: # verificar si el arreglo está lleno / check if the array is full
19         break # salir del bucle / exit the loop
20
21 print(f'la lista tiene {len(car)}') # imprimir cuántos caracteres hay en la lista
22 # / print how many characters are in the list
23
24 for i in arr: # recorrer el arreglo / loop through the array
25     if i != -1: # verificar si el valor es un número válido / check if the value is a valid number
26         c2 += 1 # incrementar el contador de números / increment the number counter
27
28 print(f'el arreglo tiene {c2}') # imprimir cuántos números hay en el arreglo / print how many numbers are in the array
29 print(car) # imprimir la lista de caracteres / print the list of characters
30 print(arr) # imprimir el arreglo de números / print the array of numbers

```

```

Escribe un dato o valor
4
Escribe un dato o valor
r
Escribe un dato o valor
8
Escribe un dato o valor
u
Escribe un dato o valor
k
Escribe un dato o valor
l
Escribe un dato o valor
b
Escribe un dato o valor
8
Escribe un dato o valor
n
la lista tiene 6
el arreglo tiene 4
['r', 'u', 'k', 'l', 'b', 'n']
[2, 4, -1, 8, -1, -1, -1, 8, -1]

```



## Opción 2 Programa 5

```

6 array = [] # arreglo para almacenar números / array to store numbers
7 lista = [] # lista para almacenar caracteres / list to store characters
8
9 while True:
10     entrada_usuario = input("Escribe un número o un carácter: ") # pedir al usuario un dato / ask the user for a value
11
12     if entrada_usuario.isdigit(): # verificar si la entrada es un número / check if the input is a number
13         array.append(entrada_usuario) # agregar el número al arreglo / add the number to the array
14         print("El número se agrego al array") # mensaje de confirmación / confirmation message
15     else:
16         lista.append(entrada_usuario) # agregar el carácter a la lista / add the character to the list
17         print("El carácter se agrego a la lista") # mensaje de confirmación / confirmation message
18
19     e_array = len(array) # contar elementos en el arreglo / count elements in the array
20     e_lista = len(lista) # contar elementos en la lista / count elements in the list
21     elementos = e_array + e_lista # total de elementos / total number of elements
22
23     if elementos >=10: # verificar si se alcanzaron 10 elementos / check if 10 elements are reached
24         break # salir del bucle / exit the loop
25
26 cantidad_array = len(array) # cantidad final de números / final count of numbers
27 cantidad_lista = len(lista) # cantidad final de caracteres / final count of characters
28
29 print(f"La cantidad de elementos dentro del array son: {cantidad_array}") # mostrar cantidad de números
30 #/ show number count
31 print(array) # mostrar el arreglo / show the array
32
33 print(f"La cantidad de elementos dentro del lista son: {cantidad_lista}") # mostrar cantidad de caracteres
34 #/ show character count
35 print(lista) # mostrar la lista / show the list

```

```

El número se agrego al array
Escribe un número o un carácter: h
El carácter se agrego a la lista
Escribe un número o un carácter: u
El carácter se agrego a la lista
Escribe un número o un carácter: i
El carácter se agrego a la lista
Escribe un número o un carácter: 9
El número se agrego al array
Escribe un número o un carácter: k
El carácter se agrego a la lista
Escribe un número o un carácter: 0
El número se agrego al array
Escribe un número o un carácter: g
El carácter se agrego a la lista
Escribe un número o un carácter: f
El carácter se agrego a la lista
La cantidad de elementos dentro del array son: 4
['1', '2', '9', '0']
La cantidad de elementos dentro del lista son: 6
['h', 'u', 'i', 'k', 'g', 'f']

```



## PROGRAMA 6

**Hacer un programa que lea 10 datos, si el dato es un número se almacenará en un arreglo, si es un carácter o caracteres se meterá a una lista, cuando finalice el programa nos mostrará cuántos elementos numéricos y cuántos caracteres hay en cada estructura. Utilizando el método principal.**

```

2 def resultados():
3     c2 = 0 # contador para elementos numéricos en el arreglo
4     #/ counter for numeric elements in the array
5     print(f'la lista tiene {len(carc)}') # mostrar la cantidad de elementos en la lista
6     #/ show the number of elements in the list
7     for i in arr: # recorrer el arreglo / loop through the array
8         if i != -1: # verificar si el valor es válido / check if the value is valid
9             c2 += 1 # incrementar el contador / increment the counter
10    print(f'el arreglo tiene {c2}') # mostrar la cantidad de números en el arreglo
11    #/ show the number of numbers in the array
12    print(carc) # mostrar la lista de caracteres / show the character list
13    print(arr) # mostrar el arreglo de números / show the array of numbers
14
15
16 def hola(): # DEFINICIÓN DE MÉTODO O FUNCIÓN / FUNCTION DEFINITION
17     c = 0 # contador para el arreglo / counter for the array
18     while True: # CICLO INFINITO / INFINITE LOOP
19         a = input('Escribe un dato: ') # solicitar dato al usuario / request input from the user
20         if a.isdigit(): # VALIDAR DIGITOS / CHECK DIGITS
21             arr[c] = int(a) # guardar número en el arreglo / store number in the array
22         elif a.isalpha(): # VALIDAR LETRAS / CHECK LETTERS
23             carc.append(a) # agrega elemento al final de la lista / add element to the end of the list
24             c += 1 # incrementar contador / increment counter
25
26         if c >= 10: # verificar si se alcanzaron 10 elementos / check if 10 elements are reached
27             break # salir del bucle / exit the loop
28     resultados() # llamar a la función resultados / call the resultados function
29
30 carc = [] # lista para caracteres / list for characters
31 arr = [-1, -1, -1, -1, -1, -1, -1, -1, -1] # arreglo inicializado con -1 / array initialized with -1
32
33 if __name__ == '__main__': # MÉTODO PRINCIPAL / MAIN METHOD
34     hola() # llamar a la función hola / call the hola function

```



```
Escribe un dato: hola
la lista tiene 1
el arreglo tiene 0
['hola']
[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
Escribe un dato: 64
la lista tiene 1
el arreglo tiene 1
['hola']
[-1, 64, -1, -1, -1, -1, -1, -1, -1, -1]
Escribe un dato: Leslie
la lista tiene 2
el arreglo tiene 1
['hola', 'Leslie']
[-1, 64, -1, -1, -1, -1, -1, -1, -1, -1]
Escribe un dato: 23
la lista tiene 2
el arreglo tiene 2
['hola', 'Leslie']
```



## PROGRAMA 7

**Hacer un programa que lea el nombre, edad y sexo de 5 personas. Estos elementos tienen que estar dentro de una lista**

```

4  def inicio(): # DEFINICIÓN DE LA FUNCIÓN / FUNCTION DEFINITION
5      c = 0 # contador de personas / counter for people
6      while True: # CICLO INFINITO / INFINITE LOOP
7          n = input('Escribe el nombre') # solicitar nombre al usuario / ask the user for the name
8          e = input('Escribe la edad') # solicitar edad al usuario / ask the user for the age
9          s = input('Escribe el sexo') # solicitar sexo al usuario / ask the user for the gender
10         lis.append([n, e, s]) # agregar los datos como lista dentro de la lista principal
11         #/ add the data as a list inside the main list
12         c += 1 # incrementar el contador / increment the counter
13         if c >= 5: # verificar si ya se ingresaron 5 personas / check if 5 people have been entered
14             break # salir del bucle / exit the loop
15     print(lis) # mostrar la lista completa / print the full list
16
17     lis = [] # lista principal para almacenar los datos de las personas / main list to store people's data
18
19     if __name__ == "__main__": # MÉTODO PRINCIPAL / MAIN METHOD
20         inicio() # llamar a la función inicio / call the inicio function

```

Escribe el nombreLeslie

Escribe la edad19

Escribe el sexoF

Escribe el nombreMario

Escribe la edad15

Escribe el sexoM

Escribe el nombreJuan

Escribe la edad35

Escribe el sexom

Escribe el nombresara

Escribe la edad21

Escribe el sexof

Escribe el nombreerick

Escribe la edad17

Escribe el sexom

[[ 'Leslie', '19', 'F'], [ 'Mario', '15', 'M'], [ 'Juan', '35', 'm'], [ 'sara', '21', 'f'], [ 'erick', '17', 'm']]



## PROGRAMA 8

**Hacer un programa que lea una cadena y muestre cuantas minúsculas, mayúsculas, números y espacios tiene**

```
5 def inicio(): # DEFINICIÓN DE LA FUNCIÓN / FUNCTION DEFINITION
6     mi = 0 # contador de letras minúsculas / counter for lowercase letters
7     may = 0 # contador de letras mayúsculas / counter for uppercase letters
8     c = 0 # contador de números / counter for numbers
9     e = 0 # contador de espacios / counter for spaces
10    numeros = "0123456789" # cadena que contiene los dígitos / string containing digits
11    cadena = input('Escribe una cadena') # solicitar al usuario que ingrese una cadena / ask the user to enter a string
12    for i in cadena: # recorrer cada carácter de la cadena / loop through each character in the string
13        if i in numeros: # verificar si el carácter es un número / check if the character is a number
14            print('es numero') # mostrar mensaje si es número / display message if it is a number
15            c += 1 # incrementar contador de números / increment number counter
16        if i == ' ': # verificar si el carácter es un espacio / check if the character is a space
17            e += 1 # incrementar contador de espacios / increment space counter
18        if ord(i) >= 97 and ord(i) <= 122: # verificar si es minúscula / check if it is lowercase
19            mi += 1 # incrementar contador de minúsculas / increment lowercase counter
20        if ord(i) >= 65 and ord(i) <= 90: # verificar si es mayúscula / check if it is uppercase
21            may += 1 # incrementar contador de mayúsculas / increment uppercase counter
22    print(f'Los numeros son: {c}\n y los espacios: {e}\n y las minusculas: {mi}\n y las mayusculas: {may}\n')
23    # mostrar resultados / display results
24
25 if __name__ == "__main__": # MÉTODO PRINCIPAL / MAIN METHOD
26     inicio() # llamar a la función inicio / call the inicio function
```

● Escribe una cadenaHola  
Los numeros son: 0  
y los espacios: 0  
y las minusculas: 3  
y las mayusculas: 1



## Opción 2 programa 8

```
3 entrada_usuario = input("Escribe una cadena: ")  
4 # solicitar al usuario que ingrese una cadena de caracteres / ask the user to enter a string  
5  
6 lista_digitos = [] # lista para almacenar números / list to store digits  
7 lista_mayus = [] # lista para almacenar letras mayúsculas / list to store uppercase letters  
8 lista_mins = [] # lista para almacenar letras minúsculas / list to store lowercase letters  
9 lista_espacios = [] # lista para almacenar espacios u otros caracteres / list to store spaces or other characters  
10  
11 for e in entrada_usuario: # recorrer cada carácter en la cadena / loop through each character in the string  
12     if e.isdigit(): # verificar si es un número / check if it is a number  
13         lista_digitos.append(e) # agregar número a la lista / add number to the list  
14     elif e.isupper(): # verificar si es mayúscula / check if it is uppercase  
15         lista_mayus.append(e) # agregar letra mayúscula a la lista / add uppercase letter to the list  
16     elif e.islower(): # verificar si es minúscula / check if it is lowercase  
17         lista_mins.append(e) # agregar letra minúscula a la lista / add lowercase letter to the list  
18     else: # si no es número ni letra / if it is neither number nor letter  
19         lista_espacios.append(e) # agregar a lista de espacios / add to spaces list  
20  
21 print(f"La palabra es: {entrada_usuario}") # mostrar la cadena ingresada / display the entered string  
22 print("La cantidad de numeros es:", len(lista_digitos)) # mostrar cantidad de números / display number count  
23 print("La cantidad de mayúsculas es:", len(lista_mayus)) # mostrar cantidad de mayúsculas / display uppercase count  
24 print("La cantidad de minúsculas es:", len(lista_mins)) # mostrar cantidad de minúsculas / display lowercase count  
25 # La comilla cuenta como espacio / The quotation mark counts as a space  
26 print("La cantidad de espacios es:", len(lista_espacios)) # mostrar cantidad de espacios / display space count  
27
```

```
Escribe una cadena: Hola Buenos Dias 7  
La palabra es: Hola Buenos Dias 7  
La cantidad de numeros es: 1  
La cantidad de mayúsculas es: 3  
La cantidad de minúsculas es: 11  
La cantidad de espacios es: 3
```



## PROGRAMA 9

**Hacer un programa que almacene cadenas de caracteres en una lista, donde las cadenas no tengan espacios, solo la primera letra esté en mayúscula, contengan obligatoriamente todas las vocales, y que el programa no termine hasta que la lista tenga cinco elementos que cumplan estas condiciones.**

```
7  def vocales(cad):
8      # Variables booleanas para cada vocal / Boolean variables for each vowel
9      ba = False
10     be = False
11     bi = False
12     bo = False
13     bu = False
14     # Verifica si la cadena contiene cada vocal (mayúscula o minúscula)
15     # Checks if the string contains each vowel (uppercase or lowercase)
16     if 'a' in cad or 'A' in cad:
17         ba = True
18     if 'e' in cad or 'E' in cad:
19         be = True
20     if 'i' in cad or 'I' in cad:
21         bi = True
22     if 'o' in cad or 'O' in cad:
23         bo = True
24     if 'u' in cad or 'U' in cad:
25         bu = True
26     # Si todas las vocales están presentes, se agrega la cadena a la lista
27     # If all vowels are present, add the string to the list
28     if ba == True and be == True and bi == True and bo == True and bu == True:
29         lista.append(cad)
30     # Imprime la lista acumulada de cadenas válidas
31     # Prints the accumulated list of valid strings
32     print(lista)
33
34
35
36  def minusculas(c1):
37      # Contador de caracteres en minúscula / Counter for lowercase characters
38      cm = 0
39      print(c1) # Imprime la cadena recibida / Prints the received string
40      # Recorre la cadena desde el segundo carácter (exceptuando el primero)
41      # Loops through the string starting from the second character
42      for i in c1[1:]:
```



```
42     for i in c1[1:]:
43         # Verifica si el carácter está en minúscula usando su valor ASCII
44         # Checks if the character is lowercase using its ASCII value
45         if ord(i) >= 97 and ord(i)<=122:
46             cm += 1
47     # Si todos menos el primero son minúsculas
48     # If all except the first are lowercase
49     if cm == len(c1)-1:
50         print(f'la cadena son minusculas excepto la primera{cm}')
51         # Llama a la función vocales (aunque aquí pasa un número, debería ser la cadena)
52         # Calls the vowels function (though here it passes a number, should be the string)
53         vocales(cm)
54     else:
55         # Si no se cumple la condición
56         # If the condition is not met
57         print('Error la cadena no cumple')
58
59
60 def inicio():
61     # Contador de caracteres distintos de espacio
62     # Counter for characters different from space
63     ce = 0
64     nc = "" # Nueva cadena para depuración / New string for debugging
65     c = input("Escribe la cadena: \n") # Pide al usuario una cadena / Asks user for a string
66     # Recorre cada carácter de la cadena
67     # Loops through each character of the string
68     for i in c:
69         if ord(i) != 32: # Si no es espacio / If it's not a space
70             ce += 1
71
72     # Si no había espacios en la cadena
73     # If there were no spaces in the string
74     if ce == len(c):
75         if c.isalpha(): # Si la cadena solo tiene letras / If string contains only letters
76             print("La cadena solo contiene letras")
```



```
76     # Revisa si son minúsculas con excepción de la primera
77     # Checks if they are lowercase except the first one
78     minusculas(c)
79 else:
80     # Si no son todas letras, revisa carácter por carácter
81     # If not all are letters, check character by character
82     for i in c:
83         # Si es un número, lo omite / If it's a number, skip it
84         if ord(i) >= 48 and ord(i) <= 57:
85             pass
86         else:
87             # Si no es número, lo agrega a la nueva cadena
88             # If not a number, add it to the new string
89             nc += i
90     print(nc) # Imprime la nueva cadena parcial / Prints the new partial string
91     minusculas(nc) # Verifica minúsculas en la nueva cadena / Checks lowercase in new string
92 else:
93     # Si contiene espacios
94     # If contains spaces
95     print('Error en la cadena')
96
97 # Lista global para almacenar las cadenas válidas
98 # Global list to store valid strings
99 lista = []
100
101 if __name__ == "__main__":
102     # Ciclo infinito hasta que haya al menos 5 cadenas válidas
103     # Infinite loop until there are at least 5 valid strings
104     while(True):
105         inicio() # Llama al inicio / Calls the main function
106         if len(lista) >=5: # Si ya hay 5 cadenas en la lista / If there are 5 strings in the list
107             break # Rompe el ciclo / Breaks the loop
108
```



## Opción 2 Programa 9

```
5 lista = []
6 # Lista global para almacenar las palabras válidas
7 # Global list to store valid words
8
9 def ejercicio(cadena):
10     # Borramos los espacios
11     # Remove spaces
12     x = cadena.replace(" ", "")
13
14     # Hacemos que toda la palabra sea en minúsculas y validamos vocales
15     # Convert the string to lowercase and validate vowels
16     y = x.lower()
17     vocales = {"a", "e", "i", "o", "u"}
18     # Conjunto de vocales
19     # Set of vowels
20     string_completa_set = set(y)
21     # Convertimos la cadena en un conjunto de caracteres únicos
22     # Convert the string into a set of unique characters
23     validar_vocales = vocales.issubset(string_completa_set)
24     # Verificamos si todas las vocales están presentes
25     # Check if all vowels are present
26
27     # Si las vocales están, transformamos la primera letra en mayúscula
28     # y agregamos la cadena formateada a la lista
29     # If vowels are present, capitalize the first letter and add the formatted string to the list
30     # En caso contrario, solicitamos otra palabra
31     # Otherwise, ask for another word
32     if validar_vocales == True:
33         string_formateada = y[0].upper() + y[1:]
34         # Primera letra en mayúscula, resto en minúscula
35         # First letter uppercase, rest lowercase
```



```
35     # First letter uppercase, rest lowercase
36     lista.append(string_formateada)
37     # Agregamos la cadena a la lista
38     # Add the string to the list
39     print(f"La palabra: {string_formateada} se agrego a la lista")
40     # Mostramos mensaje de éxito
41     # Show success message
42     print(f"La lista : {lista}")
43     # Mostramos la lista actual
44     # Show current list
45 else:
46     print(" Esta palabra no tiene todas las vocales Vuelve a escribir una que contenga todas las vocales:")
47     # Mensaje de error si faltan vocales
48     # Error message if vowels are missing
49     return None
50     # Retornamos None
51     # Return None
52
53
54 while len(lista) < 6:
55     # Bucle hasta que la lista tenga 6 elementos
56     # Loop until the list has 6 elements
57
58     # Pedimos la palabra al usuario
59     # Ask the user for a word
60     cadena = input("Ingresa Texto: ")
61
62     # Llamamos a la función para validar y agregar la palabra
63     # Call the function to validate and add the word
64     ejercicio(cadena)
65
```

```
Ingresa Texto: Hola
Esta palabra no tiene todas las vocales Vuelve a escribir una que contenga todas las vocales:
Ingresa Texto: aeiou
La palabra: Aeiou se agrego a la lista
La lista : ['Aeiou']
Ingresa Texto: █
```



## REPASO 1

**Hacer un programa que lea el nombre y el precio de un producto; el programa calculará el costo (que incluye un 12% adicional) y el precio de venta (que incluye IVA del 16%).**

```

23  for i in range(1,5): # Repite 4 veces: desde 1 hasta 4 (el 5 no se incluye)
24      | | | | # Repeat 4 times: from 1 to 4 (5 is not included)
25      # precio = 12.55 # Inicialización de ejemplo, se sobreescribe después
26      | | | | # Example initialization, overwritten later
27      nombre = input('Escribe el nombre del producto\n')
28      # Pide al usuario el nombre del producto
29      # Ask the user for the product name
30      precio = float(input("Escribe el precio del producto\n"))
31      # Pide al usuario el precio del producto y lo convierte a flotante
32      # Ask the user for the product price and convert it to float
33      costo = precio * 1.12
34      # Calcula el costo agregando 12% (sobreprecio o gastos)
35      # Calculate the cost adding 12% (overhead or expenses)
36      precioventa = costo * 1.16
37      # Calcula el precio de venta agregando IVA del 16%
38      # Calculate the selling price adding 16% VAT
39      print(f'el costo es {costo:.2f} y el precio de venta {precioventa:.2f}\n')
40      # Imprime el costo y precio de venta con 2 decimales
41      # Print the cost and selling price with 2 decimals
42      print(f'el costo es {costo} y el precio de venta {precioventa}\n')
43      # Imprime el costo y precio de venta sin limitar decimales
44      # Print the cost and selling price without limiting decimals
45
46      # Código comentado para permitir al usuario continuar o terminar
47      # Commented code to allow the user to continue or stop
48      # res = input('Deseas otro numero (s/n) \n')
49      # if res == 'n' or res == 'N':
50          #     break

```

```

Escribe el nombre del producto
tortillas
Escribe el precio del producto
15
el costo es 16.80 y el precio de venta 19.49

el costo es 16.8 y el precio de venta 19.488

Escribe el nombre del producto

```



## REPASO 2

**Hacer un programa que tenga el funcionamiento de la formula general**

```
5  a = 1
6  # Coeficiente a de la ecuación cuadrática
7  # Coefficient a of the quadratic equation
8  b = 2
9  # Coeficiente b de la ecuación cuadrática
10 # Coefficient b of the quadratic equation
11 c = -15
12 # Coeficiente c de la ecuación cuadrática
13 # Coefficient c of the quadratic equation
14 p = 0
15 # Variable para b al cuadrado (b^2)
16 # Variable for b squared (b^2)
17 m = 0
18 # Variable para 4*a*c
19 # Variable for 4*a*c
20 r = 0
21 # Variable para el discriminante
22 # Variable for the discriminant
23 ra = 0.0
24 # Variable para la raíz cuadrada del discriminante
25 # Variable for the square root of the discriminant
26 d = 0.0
27 # Variable para 2*a
28 # Variable for 2*a
29 x1 = 0.0
30 # Variable para la primera solución
31 # Variable for the first solution
32 x2 = 0.0
33 # Variable para la segunda solución
34 # Variable for the second solution
35
36 p = b**2
37 # Calculamos b al cuadrado
38 # Calculate b squared
39 m = a*c*4
```



```
40 # Calculamos 4*a*c
41 # Calculate 4*a*c
42 r = p - m
43 # Calculamos el discriminante b^2 - 4ac
44 # Calculate the discriminant b^2 - 4ac
45
46 if r > 0:
47     # Si el discriminante es positivo, existen soluciones reales
48     # If the discriminant is positive, real solutions exist
49     print('Si se puede')
50     # Indicamos que es posible calcular las soluciones
51     # Indicate that it is possible to calculate the solutions
52     ra = r **(1/2)
53     # Calculamos la raíz cuadrada del discriminante
54     # Calculate the square root of the discriminant
55     d = 2*a
56     # Calculamos 2*a para la fórmula general
57     # Calculate 2*a for the quadratic formula
58     x1 = (-b + ra)/ d
59     # Calculamos la primera solución usando la fórmula general
60     # Calculate the first solution using the quadratic formula
61     x2 = (-b - ra)/ d
62     # Calculamos la segunda solución usando la fórmula general
63     # Calculate the second solution using the quadratic formula
64     print(f'El valor de x1 es {x1:.2f} y de x2 {x2:.2f}')
65     # Mostramos los valores de las soluciones con 2 decimales
66     # Display the values of the solutions with 2 decimals
67 else:
68     # Si el discriminante no es positivo, no hay soluciones reales
69     # If the discriminant is not positive, there are no real solutions
70     print('No se puede')
71     # Indicamos que no es posible calcular soluciones reales
72     # Indicate that it is not possible to calculate real solutions
73
```

Si se puede  
El valor de x1 es 3.00 y de x2 -5.00



## REPASO 3

**Hacer un programa que lea números y defina si es decimal o entero**

```

4  def validar(a): # Definición de la función validar que recibe un parámetro 'a'
5      # Definition of the function validar that takes one parameter 'a'
6      c = 0 # Variable entera inicializada en 0
7      # Integer variable initialized to 0
8      d = 0.0 # Variable decimal inicializada en 0.0
9      # Float variable initialized to 0.0
10     try: # Intentar convertir 'a' en un número entero
11         # Try to convert 'a' into an integer
12         c = int(a)
13         print('Es un número entero sin decimales')
14         # It is an integer without decimals
15         print('Es un número entero')
16         # It is an integer
17     except ValueError: # Si ocurre un error, significa que no es entero
18         # If an error occurs, it means it's not an integer
19         try: # Intentar convertir 'a' en un número decimal (float)
20             # Try to convert 'a' into a decimal (float)
21             d = float(a)
22             print('Es un número entero con decimales')
23             # It is a number with decimals
24             print('Es un número decimal\n')
25             # It is a decimal number
26         except ValueError: # Si tampoco se puede, no es un número válido
27             # If it also fails, then it is not a valid number
28             print('No es un número válido')
29             # It is not a valid number
30
31 def leer(): # Función que pide al usuario un dato
32     # Function that asks the user for a value
33     a = input('Escribe un dato o valor\n')
34     # Read input from the user
35     validar(a)
36     # Call the function validar with the input
37 if __name__ == '__main__': # Punto de entrada principal del programa
38     # Main entry point of the program
39     leer()
40     # Call the function leer

```

Escribe un dato o valor  
2  
Es un número entero sin decimales  
Es un número entero



Escribe un dato o valor

2.3

Es un número entero con decimales

Es un número decimal



## PRACTICA REPASO 3

**Hacer un programa que lea números enteros y los almacene en una lista. Si el número ingresado no es un entero, el programa volverá a pedirlo hasta que se ingrese correctamente. Los números se seguirán almacenando hasta que el usuario decida que no quiere agregar más datos.**

```

8 def validar(a):
9     ne = 0
10    try:
11        ne = int(a) # Intenta convertir el dato a entero / Try to convert the input into an integer
12        return ne
13    except ValueError:
14        print('No es un entero') # Muestra mensaje si no es un número entero / Shows a message if it's not an integer
15    try:
16        nf = float(a) # Intenta convertir el dato a decimal / Try to convert the input into a float
17        return nf
18    except ValueError:
19        print('No es un numero con decimales') # Muestra mensaje si no es decimal / Shows a message if it's not a decimal
20    return a # Devuelve el dato original si no es número / Returns the original input if it's not a number
21
22 def leer():
23     a = input('Escribe un dato \n') # Pide al usuario un dato / Asks the user for an input
24     dato = validar(a) # Valida el dato ingresado / Validates the entered input
25     lista.append(dato) # Agrega el dato a la lista / Appends the data to the list
26
27 lista = [] # Lista vacía para guardar los datos / Empty list to store the data
28 if __name__ == '__main__':
29     while(True): # Ciclo infinito hasta que el usuario decida salir / Infinite loop until the user decides to exit
30         leer() # Llama a la función para leer y validar datos / Calls the function to read and validate data
31         res = input('Deseas otro s/n') # Pregunta si quiere continuar / Asks if the user wants to continue
32         if res == 'n' or res == 'N': # Si la respuesta es 'n' o 'N' termina / If response is 'n' or 'N', it ends
33             print(lista) # Muestra la lista de datos capturados / Prints the list of captured data
34             break # Rompe el ciclo y finaliza el programa / Breaks the loop and ends the program
35

```

```

Escribe un dato
1
Deseas otro s/nS
Escribe un dato
4
Deseas otro s/nN
[1, 4]

```



# **PARCIAL 2**

## **Recursividad**



## Programa 1

Leer 5 calificaciones y agregarlas a una lista

```
1 def inicio(num):
2     # Función que pide calificaciones y las guarda en una lista hasta tener 5
3
4     a = int(input('Escribe una calificación \n')) # Pide una calificación al usuario
5     num += 1 # Aumenta el contador de calificaciones
6     lista.append(a) # Agrega la calificación a la lista
7
8     if num >= 5: # Si ya se ingresaron 5 calificaciones
9         print(lista) # Muestra la lista completa
10    else:
11        inicio(num) # Si no, vuelve a llamar la función para pedir otra calificación
12
13
14 lista = [] # Crea una lista vacía para guardar las calificaciones
15 global num # Declara la variable num como global
16 num = 0 # Inicializa el contador en 0
17
18 if __name__ == '__main__':
19     inicio(num) # Llama a la función inicio por primera vez
```

```
Escribe una calificación
8
Escribe una calificación
9
Escribe una calificación
6
Escribe una calificación
7
Escribe una calificación
6
[8, 9, 6, 7, 6]
```



## Examen 1

Hacer un programa que lea el nombre y tres calificaciones que se agregarán con las siguientes restricciones:

Las calificaciones se agregarán en orden descendente. Una vez realizado esto, mostrará una pregunta para saber si deseas ingresar otra persona.

Si la respuesta es “sí”, la primera lista se agregará a otra lista que contenga los cuatro datos, para luego agregar a otra persona.

```
def inicio():
    # Lista temporal para guardar los datos de un solo alumno
    # Temporary list to store data for a single student
    lista1 = []

    # Se pide el nombre del alumno
    # Ask for the student's name
    nom = input('Escribe un nombre\n')

    # Se piden tres calificaciones
    # Ask for three grades
    c1 = int(input('Escribe una calificación\n'))
    c2 = int(input('Escribe una calificación\n'))
    c3 = int(input('Escribe una calificación\n'))

    # Comienza la comparación para saber cuál calificación es mayor
    # Start comparing to determine which grade is highest
    if (c1 > c2):
        if (c1 > c3):
            # Si c1 es mayor que las otras dos
            # If c1 is greater than both others
            print(f'f es mayor {c1}') # ← faltaba la f antes de la cadena
            lista1.append(nom)
            lista1.append(c1)
        if (c2 > c3):
            # Si c2 es el de en medio y c3 el menor
            # If c2 is middle and c3 is lowest
            print(f'f es el de en medio {c2}')
            print(f'f es el menor {c3}')
            lista1.append(c2)
            lista1.append(c3)
            lista2.append(lista1)
        else:
            # Si c3 es el de en medio y c2 el menor
            # If c3 is middle and c2 is lowest
```



```
# If c3 is middle and c2 is lowest
print(f'f es el de en medio {c3}')
print(f'f es el menor {c2}')
lista1.append(c3)
lista1.append(c2)
lista2.append(lista1)

else:
    # Si c3 es mayor que c1 (aunque c1 > c2)
    # If c3 is greater than c1 (even though c1 > c2)
    print(f'f es el de en medio {c1}')
    print(f'f es el mayor {c3}')
    print(f'f es el menor {c2}')
    lista1.append(c3)
    lista1.append(c1)
    lista1.append(c2)
    lista2.append(lista1)

else:
    # Si c2 es mayor o igual a c1
    # If c2 is greater than or equal to c1
    if (c2 > c3):
        # Si c2 es mayor que c3
        # If c2 is greater than c3
        print(f'f es mayor {c2}')
        lista1.append(nom)
        lista1.append(c2)
        if (c1 > c3):
            # Si c1 es el de en medio y c3 el menor
            # If c1 is middle and c3 is lowest
            print(f'f es el de en medio {c1}')
            print(f'f es el menor {c3}')
            lista1.append(c1)
            lista1.append(c3)
            lista2.append(lista1)
        else:
            # Si c3 es el de en medio y c1 el menor
            # If c3 is middle and c1 is lowest
```



```
# Si c3 es el de en medio y c1 el menor
# If c3 is middle and c1 is lowest
print(f'f es el de en medio {c3}')
print(f'f es el menor {c1}')
lista1.append(c3)
lista1.append(c1)
lista2.append(lista1)

else:
    # Si c3 es mayor que todos
    # If c3 is the highest of all
    print(f'f es el de en medio {c2}')
    print(f'f es el mayor {c3}')
    print(f'f es el menor {c1}')
    lista1.append(nom)
    lista1.append(c3)
    lista1.append(c2)
    lista1.append(c1)
    lista2.append(lista1)

# Lista general para guardar los datos de todos los alumnos
# Global list to store data for all students
lista2 = []

# Punto de entrada principal del programa
# Main program entry point
if __name__ == "__main__":
    while(True):
        # Se llama a la función para capturar datos
        # Call the function to capture data
        inicio()

        # Se pregunta si se desea continuar
        # Ask if user wants to continue
        a = input('¿Deseas agregar otro registro? s/n\n')  
A block of Python code with syntax highlighting. The code handles three cases for sorting three variables (c1, c2, c3) into two lists (lista1 and lista2). It first checks if c3 is the middle value. If so, it prints a message, appends c3 to lista1, and then adds lista1 to lista2. If c3 is the largest value, it prints messages for c2 and c3, appends them to lista1, and then adds lista1 to lista2. Finally, it initializes lista2 as an empty list, defines the main entry point, and enters a loop where it calls the inicio() function, asks if the user wants to continue, and reads input from the user.
```



```
# Lista general para guardar los datos de todos los alumnos
# Global list to store data for all students
lista2 = []

# Punto de entrada principal del programa
# Main program entry point
if __name__ == "__main__":
    while(True):
        # se llama a la función para capturar datos
        # Call the function to capture data
        inicio()

        # Se pregunta si se desea continuar
        # Ask if user wants to continue
        a = input('¿Deseas agregar otro registro? s/n\n')

        # Si la respuesta es 'n' o 'N', se termina el ciclo
        # If the answer is 'n' or 'N', end the loop
        if a == 'n' or a == 'N':
            # Se muestra la lista completa de registros
            # Show the full list of records
            print(lista2)
            break
```

```
Escribe un nombre
Leslie
Escribe una calificación
9
Escribe una calificación
8
Escribe una calificación
7
f es mayor 9
f es el de en medio 8
f es el menor 7
¿Deseas agregar otro registro? s/n
```



## Programa 2

### Validaciones

```

31     def inicio(self):
32         #Definimos el método 'inicio', encargado del flujo principal del programa
33         #Define a method called 'inicio' (start) that runs the main logic
34         self.a = input('Escribe una calificación \n')
35         #Pedimos al usuario que escriba una calificación
36         #Ask the user to input a grade and store it in 'self.a'
37         if val.validarNumeros(self.a):
38             #Usamos el método 'validarNumeros' del objeto 'val' para verificar si es un número
39             #Use the method 'validarNumeros' from the 'val' object to check if input is a number
40             self.num += 1
41             #Si la calificación es válida, aumentamos el contador en 1
42             #Increase the counter by 1 since the input was valid
43             self.lista.append(int(self.a))
44             #Convertimos la entrada a entero y la añadimos a la lista de calificaciones
45             #Convert the input to an integer and add it to the list of grades
46             if self.num >= 5:
47                 #Si ya se ingresaron 5 o más calificaciones válidas
48                 #If 5 or more valid grades have been entered
49                 print(self.lista)
50                 #Mostramos la lista completa de calificaciones
51                 #Print the complete list of grades
52                 print(f'El promedio es: {val.Promedio(self.lista)}')
53                 #Calculamos e imprimimos el promedio usando el método 'Promedio'
54                 #Calculate and print the average using 'Promedio' method

55         else:
56             self.inicio()
57             #Si aún no se completan las 5 calificaciones, volvemos a llamar al método (recursividad)
58             #If less than 5 grades, call 'inicio' again (recursion) to keep asking
59         else:
60             print('No es un numero')
61             #Si la entrada no es un número, mostramos un mensaje de error
62             #Show message if the input is not a valid number
63             self.inicio()
64             #Volvemos a llamar al método 'inicio' hasta que se ingrese un valor válido
65             #Call 'inicio' again (recursion) to ask again until a valid number is entered

66 if __name__=='__main__':
67     app = Principal()
68     #Creamos un objeto llamado 'app' de la clase 'Principal'
69     #Create an object 'app' from the 'Principal' class
70     app.inicio()
71     #Llamamos al método 'inicio' para iniciar el proceso de ingreso de calificaciones
72     #Call the 'inicio' method to start the program and begin asking for grades

```



```
3 class validacion():
4     #Definimos una clase llamada 'validacion'
5     # Define a class named 'validacion'
6
7     def __init__(self):
8         #Método constructor: se ejecuta automáticamente al crear un objeto de esta clase
9         #Constructor method: runs automatically when creating an object of this class
10        self.suma = 0
11        #Variable que guardará la suma total de las calificaciones
12        #Variable that stores the total sum of the grades
13        self.promedio = 0.0
14        #Variable que guardará el promedio (como número decimal)
15        #Variable that stores the average (as a decimal number)
16
17    def validarNumeros(self, valor):
18        #Método para validar si un valor ingresado es numérico
19        #Method to validate if the entered value is numeric
20        if valor.isdigit():
21            #El método .isdigit() devuelve True si todos los caracteres del texto son números
22            # The .isdigit() method returns True if all characters in the string are digits
23            return True
24            #Si es un número, regresa True (válido)
25            #If it is a number, return True (valid)
26        else:
27            return False
28            #Si no es un número, regresa False (inválido)
29            #If it is not a number, return False (invalid)
30
31    def Promedio(self, lista):
32        #Método para calcular el promedio de los valores dentro de una lista
33        #Method to calculate the average of the values in a list
34        for i in lista:
35            #Recorremos cada elemento 'i' dentro de la lista de calificaciones
36            # Loop through each element 'i' inside the list of grades
37            self.suma += i
38            #Sumamos cada calificación al acumulador 'suma'
39            #Add each grade to the accumulator 'suma'
40            self.promedio = self.suma / len(lista)
41            #Calculamos el promedio dividiendo la suma entre el total de elementos
42            #Calculate the average (sum divided by number of elements)
43            return self.promedio
44            #Devolvemos el promedio calculado
45            #Return the calculated average
```



```
Escribe una calificacion  
7  
Escribe una calificacion  
6  
Escribe una calificacion  
8  
Escribe una calificacion  
7  
Escribe una calificacion  
6  
[7, 6, 8, 7, 6]  
El promedio es: 6.8
```



## Programa 3

### Pedir usuario y contraseña

```
'''Creamos una ventana donde se le pide al usuarios ingresar USUARIO y CONTRASEÑA
valida si coinciden con la informacion fija admin y 12345 seguido de esto,
muestra un mensaje si es correcto o error segun el caso'''

from tkinter import *
#Importa todas las clases y funciones de Tkinter
#Imports all classes and functions from Tkinter
from tkinter import messagebox
#Importa el módulo 'messagebox' para diálogos (info/error)
#Imports the 'messagebox' module for dialog boxes (info/error)

def Ventana():
    #Define la función principal que crea la ventana
    #Defines the main function that creates the window

    def revisar():
        #Define una función interna que se ejecuta al pulsar el botón
        #Defines an inner function that runs when the button is clicked
        try:
            u = str(us.get())
            #Obtiene el texto del Entry 'us' (usuario). .get() ya devuelve str
            #Gets the text from the 'us' Entry (username). .get() already returns str
            p = str(pas.get())
            #Obtiene el texto del Entry 'pas' (contraseña).
            #Gets the text from the 'pas' Entry (password).
            if u == 'admin' and p == '12345':
                #Comprueba las credenciales hardcodeadas: usuario 'admin' y contraseña '12345'
                #Checks hardcoded credentials: username 'admin' and password '12345'
                messagebox.showinfo('Validacion','Usuario y Contraseña correcta')
                #Si coinciden, muestra un cuadro informativo con título y mensaje
                #If they match, display an info box with a title and message
            else:
                messagebox.showerror('Error','Usuario y/o Contraseña incorrecta')
                #Si no coinciden, muestra un cuadro de error con título y mensaje
                #If they don't match, display an error box with title and message
        except ValueError:
            messagebox.showerror('Error','Introduce datos')
            #Muestra un cuadro de error solicitando introducir datos válidos.
            #Shows an error box asking the user to enter valid data.
```



```

ven = Tk()
#Crea la ventana principal (instancia de Tk)
#Creates the main window (instance of Tk)
ven.title('Programa 1 con Ventanas')
#Establece el título de la ventana
#Sets the title of the window
ven.geometry('400x400')
#Define el tamaño de la ventana: 400x400 píxeles
#Defines the size of the window: 400x400 pixels

Label(ven, text='Usuario').pack(pady=10)
#Crea una etiqueta con el texto 'Usuario' dentro de 'ven' y la empaqueta (pack) con separación vertical (pady)
#Creates a label with the text 'Username' inside 'ven' and packs it with vertical spacing (pady)
us = Entry(ven)
#Crea una caja de texto (Entry) para el usuario dentro de la ventana 'ven'
#Creates a text box (Entry) for the username inside the window 'ven'
us.pack(pady=3)
#Empaque la caja de texto y agrega un pequeño espaciado vertical
#Packs the text box and adds small vertical spacing

```

```

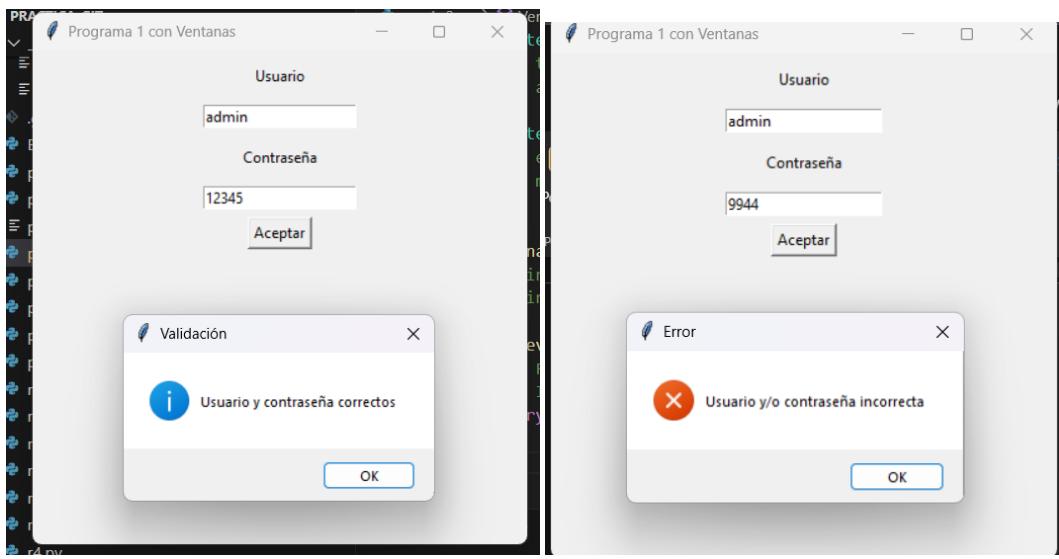
Label(ven, text='Contraseña').pack(pady=10)
#Crea otra etiqueta para 'Contraseña' y la empaqueta con separación
#Creates another label for 'Password' and packs it with spacing
pas = Entry(ven)
#Crea una caja de texto para la contraseña
#Creates a text box for the password
pas.pack(pady=3)
#Empaque la caja de la contraseña
#Packs the password text box

boton = Button(ven, text='Aceptar', command=revisar).pack(pady=3)
#Crea un botón que llama a la función 'revisar' cuando se pulsa.
#Creates a button that calls the 'revisar' function when pressed.

ven.mainloop()
#Inicia el bucle principal de la ventana (mantiene la ventana abierta y responsive)
#Starts the main window loop (keeps it open and responsive)

if __name__=='__main__':
    Ventana()
    #Llama a la función 'Ventana' para mostrar la interfaz
    #Calls the 'Ventana' function to display the interface

```





## Programa 4

Lo mismo pero con clases

```
from tkinter import *
# Importa todas las funciones de tkinter
# Imports all tkinter functions

from tkinter import messagebox
# Importa el módulo para mostrar mensajes emergentes
# Imports messagebox for pop-up dialogs

def Ventana():
    # Define la función principal
    # Defines the main function

    def revisar():
        # Función interna para validar usuario y contraseña
        # Inner function to validate username and password
        try:
            u = str(us.get())
            # Obtiene el texto del campo de usuario
            # Gets text from the username entry

            p = str(pas.get())
            # Obtiene el texto del campo de contraseña
            # Gets text from the password entry

            if u == 'admin' and p == '12345':
                # Verifica si usuario y contraseña son correctos
                # Checks if username and password are correct
                messagebox.showinfo('Validación', 'Usuario y contraseña correctos')
                # Muestra mensaje de éxito
                # Shows success message
            else:
                messagebox.showerror('Error', 'Usuario y/o contraseña incorrecta')
                # Muestra mensaje de error
                # Shows error message
        except ValueError:
            messagebox.showerror('Error', 'Introduce datos')
```



```
messagebox.showerror('Error', 'Introduce datos')
# Muestra error si ocurre un problema
# Shows error if something goes wrong

ven = Tk()
# Crea la ventana principal
# Creates the main window

ven.title('Programa 1 con Ventanas')
# Establece el título de la ventana
# Sets window title

ven.geometry('400x400')
# Define el tamaño de la ventana
# Sets window size

Label(ven, text='Usuario').pack(pady=10)
# Etiqueta "Usuario"
# Label for "Username"

us = Entry(ven)
# Campo de texto para escribir el usuario
# Entry box for username input

us.pack(pady=3)
# Empaquea el campo en la ventana
# Packs the entry in the window

Label(ven, text='Contraseña').pack(pady=10)
# Etiqueta "Contraseña"
# Label for "Password"

pas = Entry(ven)
# Campo de texto para escribir la contraseña
# Entry box for password input
```



```
" Empaque el campo en la ventana
# Packs the entry in the window

Label(ven, text='Contraseña').pack(pady=10)
# Etiqueta "Contraseña"
# Label for "Password"

pas = Entry(ven)
# Campo de texto para escribir la contraseña
# Entry box for password input

pas.pack(pady=3)
# Empaque el campo en la ventana
# Packs the entry in the window

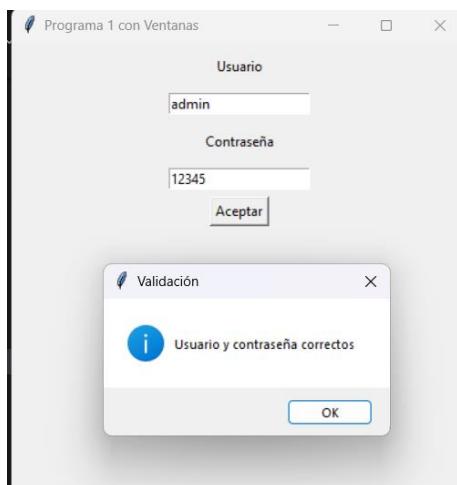
boton = Button(ven, text='Aceptar', command=revisar)
# Crea un botón que ejecuta la función revisar
# Creates a button to run revisar

boton.pack(pady=3)
# Empaque el botón en la ventana
# Packs the button in the window

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

if __name__ == "__main__":
    # Punto de entrada del programa
    # Program entry point

    Ventana()
    # Llama a la función para ejecutar la ventana
    # Calls the function to run the window
```





## Programa 5

Leer 3 calificaciones y ordenarlas de forma descendente

```
'''Hacer un programa que lea
-Nombre
-3 Calificaciones
que se agregaran a una lista con las siguientes restricciones
-Calificaciones se agregaran en orden descendente
-Mostrara una pregunta si deseas agregar otra persona, si la respuesta es correcta
-La primera lista se agregara a otra lista que contenga los 4 datos para agregar
otra persona'''

def inicio():
#Define la función 'inicio' que procesa una persona y sus 3 calificaciones
#Defines the 'inicio' function that processes one person and their 3 grades.
    lista1 = []
    #Lista temporal para guardar nombre y calificaciones ordenadas.
    #Temporary list to store name and ordered grades.
    nom = input('Escribe tu nombre: ')
    #Pide el nombre de la persona.
    #Asks for the person's name.

    cal1 = int(input('Agrega tu primera Calificacion: '))
    #Pide la primera calificación y la convierte a entero.
    #Asks for the first grade and converts it to int.
    cal2 = int(input('Agrega tu segunda Calificacion: '))
    #Pide la segunda calificación y la convierte a entero.
    #Asks for the second grade and converts it to int.
    cal3 = int(input('Agrega tu tercera Calificacion: '))
    #Pide la tercera calificación y la convierte a entero.
    #Asks for the third grade and converts it to int.

    #Comparamos las calificaciones para determinar mayor, medio y menor.
    #Compare the grades to determine largest, middle and smallest.
    if(cal1 > cal2):
        #Si cal1 es mayor que cal2.
        #If cal1 is greater than cal2.
        if(cal1 > cal3):
            #cal1 también es mayor que cal3 (cal1 es la mayor).
            #And cal1 is also greater than cal3 (cal1 is the largest).
            print(f'Es mayor {cal1}')
            #Imprime cuál es la mayor.
            #Prints which one is the largest.
            lista1.append(nom)
            #Añade el nombre a la lista temporal.
            #Adds the name to the temporary list.
```



```
lista2.append(cal1)
#Añade directamente la mayor a lista2 (inconsistencia con el resto).
#Appends the largest directly to lista2 (inconsistent with other branches).
if(cal2 > cal3):
    #Si cal2 es mayor que cal3, entonces cal2 es la del medio y cal3 la menor.
    #If cal2 is greater than cal3, then cal2 is middle and cal3 is smallest.
    print(f'Es el de en medio{cal2}')
    #Imprime la calificación del medio.
    #Prints the middle grade.
    print(f'Es el menor{cal3}')
    #Imprime la calificación menor.
    #Prints the smallest grade.
    lista1.append(cal2)
    #Añade cal2 (medio) a la lista temporal.
    #Adds cal2 (middle) to the temporary list.
    lista1.append(cal3)
    #Añade cal3 (menor) a la lista temporal.
    #Adds cal3 (smallest) to the temporary list.
    lista2.append(lista1)
    #Añade la lista temporal completa a la lista global lista2.
    #Appends the full temporary list to the global lista2.
```



```
else:  
    #Si cal3 >= cal2, entonces cal3 es la del medio y cal2 la menor.  
    #If cal3 >= cal2, then cal3 is middle and cal2 is smallest.  
        print(f'Es el de en medio{cal3}')  
        #Imprime la calificación del medio.  
        #Prints the middle grade.  
        print(f'Es el menor {cal2}')  
        #Imprime la calificación menor.  
        #Prints the smallest grade.  
  
        lista1.append(cal3)  
        #Añade cal3 (medio) a la lista temporal.  
        #Adds cal3 (middle) to the temporary list.  
        lista1.append(cal2)  
        #Añade cal2 (menor) a la lista temporal.  
        #Adds cal2 (smallest) to the temporary list.  
        lista2.append(lista1)  
        #Añade la lista temporal completa a lista2.  
        #Appends the full temporary list to lista2.
```

```
#Si cal1 > cal2 pero cal1 <= cal3, entonces cal3 es la mayor.  
#If cal1 > cal2 but cal1 <= cal3, then cal3 is the largest.  
else:  
    print(f'Es el de en medio {cal1}')  
    #Aquí cal1 sería la del medio.  
    #Here cal1 would be the middle grade.  
    print(f'Es mayor {cal3}')  
    #Imprime la mayor (cal3).  
    #Prints the largest (cal3).  
    print(f'Es el menor {cal2}')  
    #Imprime la menor (cal2).  
    #Prints the smallest (cal2).
```



```
lista1.append(nom)
#Añade el nombre a la lista temporal.
#Adds the name to the temporary list.
lista1.append(cal3)
#Añade la mayor (cal3) primero.
#Adds the largest (cal3) first.
lista1.append(cal1)
#Añade la del medio (cal1).
#Adds the middle grade (cal1).
lista1.append(cal2)
#Añade la menor (cal2).
#Adds the smallest (cal2).
lista2.append(lista1)
#Añade la lista temporal ordenada a lista2.
#Appends the ordered temporary list to lista2.
```



```
#Caso en que cal1 <= cal2 (cal2 podría ser mayor que cal1).
#Case where cal1 <= cal2 (cal2 might be greater than cal1).
else:
    if(cal2 > cal3):
        #Si cal2 es mayor que cal3, entonces cal2 es la mayor.
        #If cal2 is greater than cal3, then cal2 is the largest.
        print(f'Es mayor {cal2}')
        #Imprime la mayor.
        #Prints the largest.
        lista1.append(nom)
        #Añade el nombre a la lista temporal.
        #Adds the name to the temporary list.
        lista1.append(cal2)
        #Añade la mayor (cal2).
        #Adds the largest (cal2).
        if(cal1 > cal3):
            #Si cal1 > cal3, entonces cal1 es la del medio y cal3 la menor.
            #If cal1 > cal3, then cal1 is middle and cal3 is smallest.
            print(f'Es el de en medio {cal1}')
            #Imprime la del medio.
            #Prints the middle grade.
            print(f'Es el menor {cal3}')
            #Imprime la menor.
            #Prints the smallest.
```

```
lista1.append(cal1)
#Añade cal1 (medio) a la lista temporal.
#Adds cal1 (middle) to the temporary list.
lista1.append(cal3)
#Añade cal3 (menor) a la lista temporal.
#Adds cal3 (smallest) to the temporary list.
lista2.append(lista1)
#Añade la lista temporal a lista2.
#Appends the temporary list to lista2.
```



```
#Si cal3 >= cal1, entonces cal3 es la del medio y cal1 la menor.  
#If cal3 >= cal1, then cal3 is middle and cal1 is smallest.  
else:  
    print(f'Es el de en medio {cal3}')  
    #Imprime la del medio.  
    #Prints the middle grade.  
    print(f'Es el menor {cal1}')  
    #Imprime la menor.  
    #Prints the smallest.  
  
    lista1.append(cal3)  
    #Añade cal3 (medio).  
    #Adds cal3 (middle).  
    lista1.append(cal1)  
    #Añade cal1 (menor).  
    #Adds cal1 (smallest).  
    lista2.append(lista1)  
    #Añade la lista temporal a lista2.  
    #Appends the temporary list to lista2.
```



```
#Aquí cal3 >= cal2 (cal3 es la mayor).  
#Here cal3 >= cal2 (cal3 is the largest).  
else:  
    print(f'Es el de en medio {cal2}')  
    #Imprime la del medio (cal2).  
    #Prints the middle grade (cal2).  
    print(f'Es el mayor {cal3}')  
    #Imprime la mayor (cal3).  
    #Prints the largest (cal3).  
    print(f'Es el menor {cal1}')  
    #Imprime la menor (cal1).  
    #Prints the smallest (cal1).
```



```
lista1.append(nom)
#Añade el nombre.
#Adds the name.

lista1.append(cal3)
#Añade la mayor primero.
#Adds the largest first.

lista1.append(cal2)
#Añade la del medio.
#Adds the middle grade.

lista1.append(cal1)
#Añade la menor.
#Adds the smallest.

lista2.append(lista1)
#Guarda la lista ordenada en lista2.
#Stores the ordered list in lista2.
```

```
lista2 = []
if __name__=='__main__':
    while(True):
        inicio()
        a = input('Deseas otra persona s/n: ')
        if a =='n' or a =='N':
            print(lista2)
            break
```



```
Escribe tu nombre: Leslie
Agrega tu primera Calificacion: 8
Agrega tu segunda Calificacion: 9
Agrega tu tercera Calificacion: 7
Es mayor (9)
Es el de en medio (8)
Es el menor (7)
Deseas otra persona s/n: █
```



## Programa 6

Ingresar 4 calificaciones y sacar el promedio total y agregarlas en una lista

```
'''Ingresas 4 calificaciones, en boton de Promedio se agrega a la lista
la suma del total de las 4 calificaciones, ingresas otras calificaciones
y se agrega a la lista, ademas suma el promedio general de todas
las calificaciones agregadas en la lista'''

from tkinter import *
#Importa todas las funciones de la librería Tkinter (para crear interfaces gráficas).
#Imports all functions from the Tkinter library (for creating graphical interfaces).
from tkinter import messagebox
#Importa el módulo messagebox para mostrar cuadros de mensaje.
#Imports the messagebox module to display message dialogs.
```

```
class Principal():

#Define la clase Principal, que controlará toda la ventana y sus funciones.
#Defines the Principal class, which controls the window and its functions.

    def __init__(self):
        #Método constructor: se ejecuta al crear el objeto de la clase.
        #Constructor method: runs when the class object is created.
        self.ven = Tk()
        #Crea la ventana principal usando Tkinter.
        #Creates the main window using Tkinter.
        self.ven.title('Programa 5 con Ventanas')
        #Establece el título de la ventana.
        #Sets the title of the window.
        self.ven.geometry('600x400')
        #Define el tamaño de la ventana (600x400 píxeles).
        #Sets the size of the window (600x400 pixels).
        self.lista = []
        #Crea una lista vacía para almacenar los promedios individuales.
        #Creates an empty list to store individual averages.
        self.inicio()
        #Llama al método inicio() que crea los elementos visuales de la ventana.
        #Calls the inicio() method that builds the visual components of the window.
```



```
def sumar(self):
#Define una función que suma todos los valores guardados en self.lista.
#Defines a function that sums all values stored in self.lista.
    s = 0
    #Inicializa una variable acumuladora.
    #Initializes an accumulator variable.
    for i in self.lista:
        #Recorre cada elemento de la lista.
        #Loops through each element of the list
        s += i
        #Suma cada número al total acumulado.
        #Adds each number to the accumulated total
    return s
    #Devuelve la suma total.
    #Returns the total sum.
```

```
def promediar(self):
#Función que calcula el promedio de cuatro números y actualiza las etiquetas.
#Function that calculates the average of four numbers and updates the labels.
try:
    #Intenta ejecutar el siguiente bloque, en caso de error se usa "except".
    #Tries to execute the following block; if error occurs, "except" handles it.
    a = float(self.n1.get())
    #Convierte el texto del primer Entry a número decimal.
    #Converts text from first Entry to a decimal number.
    b = float(self.n2.get())
    #Segundo número.
    #Second number.
    c = float(self.n3.get())
    #Tercer número.
    #Third number.
    d = float(self.n4.get())
    #Cuarto número.
    #Fourth number.
    pro=(a+b+c+d)/4
    #Calcula el promedio de los cuatro números.
    #Calculates the average of the four numbers.
```



```
self.16.config(text=str(pro))
#Muestra el promedio en la etiqueta 16.
#Displays the average in label 16.
self.lista.append(pro)
#Agrega el promedio calculado a la lista de promedios.
#Adds the calculated average to the list of averages.
self.17.config(text=str(self.lista))
#Muestra la lista completa de promedios en 17.
#Displays the full list of averages in 17.
self.n1.delete(0,END)
self.n2.delete(0,END)
self.n3.delete(0,END)
self.n4.delete(0,END)
#Limpia las cajas de texto después de calcular el promedio.
#Clears the text boxes after calculating the average.
suma = self.sumar()
#Llama al método que suma todos los promedios almacenados.
#Calls the method that sums all stored averages.
p = suma / len(self.lista)
#Calcula el promedio general de todos los promedios ingresados.
#Calculates the general average of all entered averages.
self.18.config(text=f'Promedio General: {str(p)}')
#Muestra el promedio general en 18.
#Displays the general average in 18.
```



```
except ValueError:  
    messagebox.showerror("Error", "Algun dato no es numero")  
    #Muestra mensaje de error.  
    #Shows error message.  
    self.n1.delete(0,END)  
    self.n2.delete(0,END)  
    self.n3.delete(0,END)  
    self.n4.delete(0,END)  
    #Limpia las cajas para que el usuario vuelva a ingresar valores.  
    #Clears the boxes so the user can re-enter values.  
  
def salir(self):  
    self.ven.destroy()  
    #Cierra la ventana.  
    #Closes the window.  
  
def inicio(self):  
    #Crea todos los elementos visuales (etiquetas, botones, cajas de texto).  
    #Creates all visual elements (labels, buttons, text boxes).  
    l1 = Label(self.ven, text="Escribe un numero: ").place(y=10, x=20)  
    #Etiqueta para el primer número.  
    #Label for the first number  
    l2 = Label(self.ven, text="Escribe un numero: ").place(y=50, x=20)  
    #Etiqueta para el segundo número.  
    #Label for the second number.  
    self.n1 = Entry(self.ven)  
    #Caja de texto para el primer número.  
    #Text box for the first number.  
    self.n1.place(y=10, x=130)  
    #Ubica la caja en la ventana.  
    #Places the box in the window  
    self.n2 = Entry(self.ven)  
    #Caja para el segundo número.  
    #Box for the second number.  
    self.n2.place(y=50, x=130)  
    #Coloca la caja en su posición.  
    #Positions the box.
```



```
l3 = Label(self.ven, text="Escribe un numero: ").place(y=90, x=20)
#Tercer número.
#Third number.
l4 = Label(self.ven, text="Escribe un numero: ").place(y=130, x=20)
#Cuarto número.
#Fourth number.
self.n3 = Entry(self.ven)
self.n3.place(y=90, x=130)
self.n4 = Entry(self.ven)
self.n4.place(y=130, x=130)

l5 = Label(self.ven, text="Promedio: ").place(y=150, x=130)
#Etiqueta para mostrar el promedio individual.
#Label to display the individual average.
self.l6 = Label(self.ven, text="0.0")
#Valor inicial del promedio.
#Initial value of the average.
self.l6.place(y=150, x=200)

b1 = Button(self.ven, text="Promedio", command=self.promediar).place(y=50, x=300)
#Botón que calcula el promedio individual.
#Button that calculates the individual average.
b2 = Button(self.ven, text="Salir", command=self.salir).place(y=90, x=300)
#Botón que cierra el programa.
#Button that closes the program
```



```
self.l7 = Label(self.ven, text="[]")
#Muestra la lista de promedios individuales.
#Displays the list of individual averages.
self.l7.place(y=170, x=200)

self.l8 = Label(self.ven, text="Promedio General: 0.0")
#Muestra el promedio general acumulado.
#Displays the accumulated general average.
self.l8.place(y=190, x=130)

self.ven.mainloop()
#Inicia el bucle principal de la interfaz (mantiene la ventana abierta)
#Starts the main GUI loop (keeps the window running).

if __name__=='__main__':
    app = Principal()
    #Crea una instancia de la clase Principal y lanza la aplicación.
    #Creates an instance of the Principal class and launches the app.
```



Escribe otro numero

Escribe un numero

**Agregar** **Mayor** **Menor** **Salir**

El mayo... X

i 5

OK

[4 5 6]

This image shows a screenshot of a programming application window titled "MANUAL DE PROGRAMACION". At the top, there is a message "Escribe otro numero". Below it, there is a label "Escribe un numero" followed by an empty text input field. Underneath the input field is another empty text input field. To the left of these fields are four buttons: "Agregar", "Mayor", "Menor", and "Salir". On the right side of the window, there is a modal dialog box titled "El mayo..." with an "X" button in the top right corner. The dialog box contains an information icon (a blue circle with a white 'i') and the number "5". At the bottom right of the dialog is a blue-outlined "OK" button. At the very bottom of the application window, there is a horizontal bar containing the text "[4 5 6]".



## Programa 8

Ingresar números random y determinar cuál es el mayor, menor y el de en medio

```
'''Ingresa dos numeros random, se ingresan a una lista y existen dos botones
numero mayor y menor lo cual te sale en un mensaje cual es el numero mayor
y el numero menor PLACE'''
```

```
from tkinter import *
from tkinter import messagebox
import random
#Importa el módulo random para generar números aleatorios
#Imports random library to generate random numbers
```

```
class Principal():
    #Define una clase llamada Principal / Defines a class called Principal
    def __init__(self):
        #Método constructor que se ejecuta al crear el objeto
        #Constructor method that runs when the object is created
        self.ven = Tk()
        #Crea la ventana principal / Creates the main window
        self.ven.title('Programa 7 con Ventanas PLACE')
        self.ven.geometry('550x250')

        self.a = 0
        #Variable para el primer número / Variable for the first number
        self.b = 0
        #Variable para el segundo número / Variable for the second number
        self.lista = []
        #Lista para guardar los números / List to store numbers
        self.aux1 = 0
        #Variable auxiliar para encontrar el número mayor
        #Auxiliary variable to find the largest number
        self.aux2 = 0
        #Variable auxiliar para encontrar el número menor
        #Auxiliary variable to find the smallest number
        self.cont = 0
        #Contador usado en la función mayor / Counter used in the “mayor” (max) function
```



```
def inicio(self):
    #Método que construye la interfaz / Method that builds the interface
    l = Label(self.ven, text="Programa 10")
    l.place(x=150, y=20)
    #Crea una etiqueta / Creates a label

    l2 = Label(self.ven, text="Escribe un numero")
    l2.place(x=100, y=50)
    #Crea otra etiqueta / Creates another label

    self.n1 = Entry(self.ven)
    self.n1.place(x=100, y=75)
    #Crea una caja de texto / Creates an input box

    l3 = Label(self.ven, text="Escribe otro numero")
    l3.place(x=250, y=50)
    #Crea una tercera etiqueta / Creates a third label

    self.n2 = Entry(self.ven)
    self.n2.place(x=250, y=75)
    #Crea otra caja de texto / Creates another input box

    b1 = Button(self.ven, text="Aregar", command=self.agregar)
    b1.place(x=110, y=110)
    #Coloca el botón / Places the button
```



```
b2 = Button(self.ven, text="Mayor", command=self.mayor)
b2.place(x=180, y=110)
#Botón para mostrar el número mayor / Button to show the largest number

b3 = Button(self.ven, text="Menor", command=self.menor)
b3.place(x=250, y=110)
# Botón para mostrar el número menor / Button to show the smallest number
# (parameter) self: Self@Principal

b4 = Button(self.ven, text="Salir", command=self.salir)
b4.place(x=310, y=110, width=50)
#Botón para salir / Exit button

self.listaElementos = Label(self.ven, text=" ")
self.listaElementos.place(x=150, y=180)
#Etiqueta para mostrar los elementos de la lista / Label to show the list elements
self.listview = Listbox(self.ven, height=10,
width=15, bg='black',
activestyle='dotbox', fg='white')
#Caja de lista para mostrar los números agregados / Listbox to display added numbers
self.listview.place(x=410, y=20)
#Coloca la lista / Places the listbox

self.ven.mainloop()
#Mantiene la ventana abierta / Keeps the window open
```



```
def mayor(self):
#Función para encontrar el número mayor / Function to find the largest number
    if len(self.lista) > 0:
        #Verifica si la lista tiene elementos / Checks if the list has elements
        if self.aux1 < self.lista[self.cont]:
            #Si el número actual es mayor que aux1 / If the current number is greater than aux1
            self.aux1 = self.lista[self.cont]
            #Actualiza el número mayor / Updates the largest number
        self.cont += 1
        #Aumenta el contador / Increases the counter
        if len(self.lista)-1 < self.cont:
            #Si ya recorrió toda la lista / If the entire list has been checked
            print(f'El mayor es {self.aux1}')
            #Muestra en consola / Prints to console
            messagebox.showinfo('El mayor ', self.aux1)
            #Muestra en ventana emergente / Shows in a popup window
            self.cont = 0
            #Reinicia el contador / Resets the counter
        else:
            return self.mayor()
            #Llama la función de nuevo (recursividad) / Calls the function again (recursion)

    else:
        messagebox.showerror("Error","La lista esta vacia")
        #Error si no hay datos / Error if the list is empty
```

```
def menor(self):
#Función para encontrar el número menor / Function to find the smallest number
    if len(self.lista) > 0:
        #Verifica si hay elementos / Checks if there are elements
        for i in self.lista:
            #Recorre la lista / Iterates through the list
            if self.aux2 > i:
                #Si el valor actual es menor / If the current value is smaller
                self.aux2 = i
                #Actualiza el número menor / Updates the smallest number
        print(f'El menor es {self.aux2}')
        #Muestra el resultado / Prints the result
        messagebox.showinfo('El menor ', self.aux2)
        #Muestra ventana con el resultado / Shows popup with the result
    else:
        messagebox.showerror("Error","La lista esta vacia")
        #Muestra error si no hay datos / Shows error if list is empty
```



```
def agregar(self):
#Función para agregar números aleatorios / Function to add random numbers
try:
    valor = random.randint(1,100)
    #Genera un número aleatorio / Generates a random number
    self.lista.append(valor)
    #Lo agrega a la lista / Adds it to the list
    self.listview.insert(self.listview.size()+1,valor )
    #Lo muestra en el Listbox / Displays it in the listbox

    print(self.lista)
    #Muestra la lista en consola / Prints the list in console
    self.aux2 = self.lista[0]
    #Inicializa el menor con el primer número / Initializes smallest with first number

    self.listaElementos.config(text=f'{self.lista}')
    #Actualiza el Label / Updates the label

except ValueError:
    messagebox.showerror("Error","Algun dato no es numero")
    #Error si ocurre un problema / Shows error if something goes wrong
    return self.agregar
    #Retorna la función / Returns the function
```

```
def salir(self):
#Función para cerrar la ventana / Function to close the window
    self.ven.destroy()
    # Destruye la ventana principal / Destroys the main window

if __name__=='__main__':
    app = Principal()
    #Crea un objeto de la clase Principal / Creates an object of the Principal class
    app.inicio()
    #Llama al método inicio() para mostrar la ventana / Calls inicio() to display the window
```



Programa 6 con Ventanas

Escribe otro numero

Escribe un numero

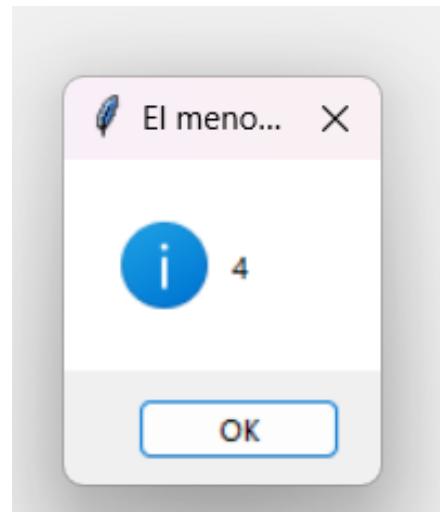
Programa 6 con Ventanas

Escribe otro numero

Escribe un numero

Escribe otro numero

Escribe un numero





## Programa 9

Validar números, letras y signos

```
'''REPASO'''
from tkinter import *
from tkinter import messagebox
from validarp9 import Validar
#Importa la clase Validar desde otro archivo / Imports the Validar class from another file

class Principal():
    def __init__(self):
        self.ventana = Tk()
        self.ventana.geometry("400x200")
        self.lista = []
        self.valid = Validar()

def inicio(self):
    Label(self.ventana, text="Programa de python con TKInter").place(x=100, y=20)
    Label(self.ventana, text="Escribe un dato").place(x=50, y=50)
    self.dato = Entry(self.ventana)
    self.dato.place(x=150, y=50, height=20)
    Button(self.ventana, text="Validar", command=self.ValidarDatos).place(x=150, y=90, width=120)
    self.mostrar = Label(self.ventana, text=" ")
    self.mostrar.place(x=20, y=150)

self.ventana.mainloop()

def ValidarDatos(self):
    val = self.dato.get()
    #Obtiene el dato de la caja de texto / Gets the value from the entry widget
    if val != "":
        #Verifica que no esté vacía / Checks that it is not empty
        self.Revisar(val)
        #Llama a Revisar para imprimir el valor / Calls Revisar to print the value
        self.lista.append(val)
        #Agrega el valor a la lista / Appends the value to the list
        self.dato.delete(0,END)
        #Limpia la caja de texto / Clears the entry widget
        self.mostrar.config(text=f'{self.lista}')
        #Actualiza el label con la lista / Updates the label with the list
```



```
'''TIPOS DE VALIDACION'''
#respuesta = self.valid.ValidarAscii(val)
respuesta = self.valid.ValidarConError(val)
#respuesta = self.valid.ValidarConString(val)

messagebox.showinfo("Validar datos",f'El datos es: {respuesta}')
#Muestra el resultado de la validación / Shows the validation result
else:
    messagebox.showerror("Error","Caja de texto esta vacia")

def Revisar(self, v):
    #Función auxiliar para imprimir en consola / Helper function to print to console
    print(v)
    #Imprime el dato / Prints the value

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

class Validar():
    def __init__(self):
        self.index = 0

    '''LETRAS MAYUSCULAS Y NUMEROS'''
    def ValidarAscii(self, valor):
        con = 0
        con2 = 0
        for i in valor:
            if ord(i) >= 48 and ord(i) <=57:
                con += 1
            if (ord(i) >=65 and ord(i) <=90) or(ord(i) >=197 and ord(i) <=122):
                con2 += 1
        if con == len(valor):
            return "Numeros"
        elif con2 == len(valor):
            return "Letras"
        else:
            return "Letras y Numeros"
```



```
'''NUMEROS DECIMALES Y ENTEROS'''

def ValidarConError(self, valor):
    a = 0
    b = 0.0
    try:
        a= int(valor)
        return "numeros"
    except ValueError:
        try:
            b = float(valor)
            return "Es numero real o decimales"
        except ValueError:
            return "letras o numeros"
```

```
'''VALIDAR CORREO'''

def ValidarConString(self, valor):
    print(valor)
    if self.index < len(valor):
        if valor[self.index] == "@":
            return "Si es un correo"
        else:
            if self.index < len(valor):
                self.index += 1
                return self.ValidarConString(valor)
            else:
                return "No es un correo"
    else:
        return "No es un correo"
```



Simulacro examen

## Simulacro Examen

Elemento: 0.

hola  6

Error

Algun campo no es correcto.

OK

# PARCIAL 3



## Estructuras Lineales

### Programa 1

Este programa en **Tkinter (Python)** permite **insertar, eliminar y ordenar números** en una lista visual (Listbox).

El usuario puede elegir entre trabajar con una **Pila** (LIFO) o una **Cola** (FIFO) mediante **radio buttons**.

Cada número ingresado pasa por una **validación**:

- Solo se aceptan **números**.
- Solo se permiten **dos dígitos** por número.

Una vez validados, los números se agregan a la lista.

También se puede eliminar elementos según el modo elegido:

- **Pilas (LIFO)**: se elimina el último elemento agregado.
- **Colas (FIFO)**: se elimina el primer elemento agregado.



```
# buebuja y seleccion de ordenamiento conceptos
# Bubble and selection sort concepts
# agregar 2 radio button arriba de ordenar abajo de radio button eliminar y
# add 2 radio buttons above "ordenar" and below "eliminar" button

from tkinter import *
from tkinter import messagebox
from Validaciones import Validar
import numpy as np

class Principal():
    def __init__(self):
        self.val = Validar() # Inicializa la clase de validaciones / Initialize validation class
        self.ven = Tk()       # Crea la ventana principal / Create main window

        # Configuración de tamaño y centrado / Set window size and center it
        ancho = 320
        alto = 210
        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}')

        # Lista vacía donde se guardan los datos / Empty list to store data
        self.lis = []

    def validarCaja(self):
        # Obtiene el valor del cuadro de texto / Get value from entry box
        valor = self.dato.get()

        # Verifica si el valor es un número y cumple con las reglas / Check if value is numeric and valid
        if (self.val.ValidarNumeros(valor)):
            if (self.val.ValidarEntradas(valor)):
                # Agrega el valor a la lista visual / Add value to visual list
                self.lista.insert(self.lista.size()+1, valor)
                self.dato.delete(0,END)
```



```
else:
    # Error si no cumple con el límite de dígitos / Error if exceeds digit limit
    messagebox.showerror("Error","Solo se permite dos digitos")
    self.dato.delete(0,END)
else:
    # Error si no es un número / Error if not numeric
    messagebox.showerror("Error","No son numeros")
    self.dato.delete(0,END)

# Actualiza el contador de elementos / Update element counter
self.label.config(text=f'Elementos en la lista: {str(self.lista.size())}')

def eliminarDato(self):
    # Verifica si la lista está vacía / Check if list is empty
    if self.lista.size() <= 0:
        messagebox.showerror("Error","La lista esta vacia")
        return

    # Elimina según el modo seleccionado (Pila o Cola) / Delete according to mode (Stack or Queue)
    if self.modo.get() == 'Pilas':
        # Último que entra, primero que sale (LIFO) / Last in, first out
        self.lista.delete(self.lista.size()-1)
    else:
        # Primero que entra, primero que sale (FIFO) / First in, first out
        self.lista.delete(0)

    # Actualiza contador de elementos / Update counter
    self.label.config(text=f'Elementos en la lista: {str(self.lista.size())}')

def ordenar(self):
    # Obtiene todos los elementos de la lista / Get all elements from list
    self.lis = list(self.lista.get(0,END))

    # Verifica si la lista está vacía / Check if list is empty
    if len(self.lista) <= 0:
```



```
    messagebox.showerror("Error", "Lista Vacia")
else:
    # Algoritmo de ordenamiento burbuja / Bubble sort algorithm
    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

    # Muestra la lista ordenada en consola / Show sorted list in console
    print(self.lis)

    # Limpia y actualiza la lista visual / Clear and update visual list
    self.lista.delete(0,END)
    for i in self.lis:
        self.lista.insert(self.lista.size()+1, i)

# Código comentado de ordenamiento por selección (alternativo)
...
# SELECCION / SELECTION SORT
p = 0
for i in range(0, len(self.lis)):
    aux = int(self.lis[i])
    p = i
    for x in range(i, len(self.lis)):
        if aux < int(self.lis[x]):
            aux = int(self.lis[x])
            p = x
    self.lis[p] = self.lis[i]
    self.lis[i] = str(aux)
print(self.lis)
self.lista.delete(0,END)
for i in self.lis:
    self.lista.insert(self.lista.size()+1, str(i))
```



```
def inicio(self):
    # Caja de texto para ingresar números / Entry box to input numbers
    self.dato = Entry(self.ven)
    self.dato.place(x=50, y=10)

    # Radio buttons para seleccionar modo (Pilas o Colas) / Radio buttons to select mode
    self.modo = StringVar(value="Pilas")
    Radiobutton(self.ven, text="Pilas", variable=self.modo, value="Pilas").place(x=50, y=40)
    Radiobutton(self.ven, text="Colas", variable=self.modo, value="Colas").place(x=100, y=40)

    # Botones principales / Main buttons
    Button(self.ven, text="Validar", command=self.validarCaja, width=10).place(x=100, y=100)
    Button(self.ven, text="Eliminar", command=self.eliminarDatos, width=10).place(x=100, y=130)
    Button(self.ven, text="Ordenar", command=self.ordenar, width=10).place(x=100, y=160)

    # Etiqueta que muestra cantidad de elementos / Label showing number of elements
    self.label = Label(text="Número")
    self.label.place(x=5, y=70)

    # Listbox para mostrar los datos / Listbox to display data
    self.lista= Listbox(self.ven, height=10, width=10, bg="white", font=("Helvetica",12))
    self.lista.place(x=190, y=10)

    # Inicia la ventana principal / Start main window loop
    self.ven.mainloop()

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



## Programa 1 Validaciones

```

class Validar():
    def __init__(self):
        # Contador para recorrer los caracteres uno por uno / Counter to iterate characters one by one
        self.con = 0

    def ValidarNumeros(self, num):
        # Si el contador llega al final del texto, regresa True / If counter reaches the end, return True
        if self.con >= len(num):
            self.con = 0
            return True
        # Verifica si el carácter actual es un número (código ASCII del 0 al 9) / Checks if current character is a digit
        if ord(num[self.con]) >= 47 and ord(num[self.con]) <= 58:
            self.con += 1
            return self.ValidarNumeros(num) # Llamada recursiva / Recursive call
        else:
            self.con = 0
            return False # Si encuentra un carácter no numérico / If non-numeric character found

    def ValidarLetra(self, dato):
        # Verifica si el primer carácter es una letra mayúscula (A-Z) / Checks if first character is uppercase (A-Z)
        if ord(dato[0]) >= 65 and (dato[0]) <= 90:
            return True
        else:
            return False

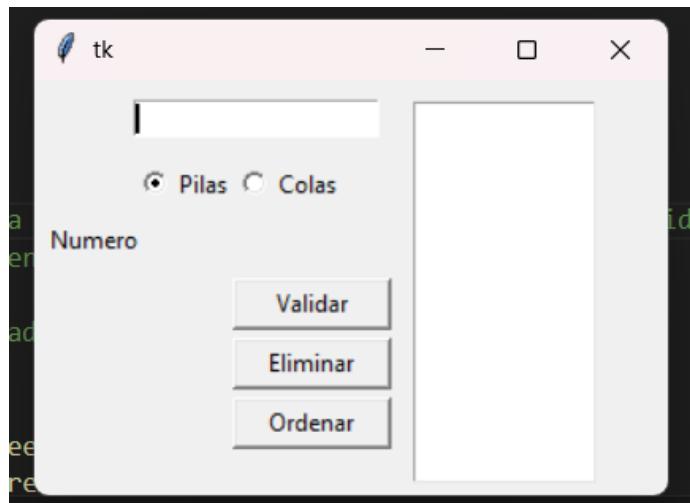
    def ValidarEntradas(self, dato):
        # Si el dato está vacío / If the input is empty
        if dato == "":
            return False
        # Si el dato tiene longitud igual a 2 / If input length equals 2
        if len(dato) == 2:
            return True
        else:
            return False

```

```

def ValidarNombre(self, nom):
    # Contador de caracteres válidos / Counter for valid characters
    c = 0
    for i in nom:
        # Verifica si el carácter está entre a-z o A-Z / Checks if character is between a-z or A-Z
        if (ord(i) >= 97 and i <= 122) or (ord(i) >= 97 and i <= 122) or ()�:
            c += 1
    # Si todos los caracteres son válidos / If all characters are valid
    if c == len(nom):
        return True
    else:
        return False

```



## Programa 2

Solicita nombre, apellidos, día, mes y año de nacimiento en cajas de texto separadas.

Al presionar el botón, genera el RFC de la persona siguiendo una estructura básica.

Los RFC generados se almacenan en un Listbox y pueden eliminarse usando pilas (LIFO) o colas (FIFO), según el modo seleccionado. Incluye validaciones de letras, números y fecha.



```
#Hacer un programa que le nombre, apellido paterno y materno en tres cajas separadas, ademas leer dia, mes y año en 3 caja
#de texto separadas. si presiona un boton se agregara a un listbox el rfc de la persona, ademas tendra
#dos botones para eliminar elementos del list box mediante pilas o colas
from tkinter import *
from tkinter import messagebox, ttk
from valRFC import Validar
from collections import deque

class Principal:
    def __init__(self):
        # Inicializa la ventana principal y variables / Initialize main window and variables
        self.val = Validar()
        self.ven = Tk()
        self.ven.title("Generador de RFC")

        # Configuración del tamaño de ventana / Window size setup
        ancho = 550
        alto = 400
        pantalla_ancho = self.ven.winfo_screenwidth()
        pantalla_alto = self.ven.winfo_screenheight()
        x = (pantalla_ancho // 2) - (ancho // 2)
        y = (pantalla_alto // 2) - (alto // 2)
        self.ven.geometry(f"{ancho}x{alto}+{x}+{y-100}")

        # Estructuras de datos / Data structures
        self.pila = []          # Pila (LIFO) / Stack
        self.cola = deque()      # Cola (FIFO) / Queue
        self.modo = StringVar(value="PILA") # Modo de eliminación por defecto / Default deletion mode

    def generarRFC(self):
        # Lee los datos de las cajas de texto / Reads data from text fields
        nombre = self.nom.get().strip()
        ap_pat = self.ap.get().strip()
        ap_mat = self.am.get().strip()
        dia = self.dia.get().strip()
        mes = self.mes.get().strip()
        anio = self.anio.get().strip()
```



```
# Validaciones de entrada / Input validations
if not (nombre and ap_pat and ap_mat and dia and mes and anio):
    messagebox.showerror("Error", "Faltan datos / Missing data")
    return

if not (self.val.ValidarLetras(nombre) and self.val.ValidarLetras(ap_pat) and self.val.ValidarLetras(ap_mat)):
    messagebox.showerror("Error", "Nombre y apellidos deben contener solo letras / Name and surnames must contain letters only")
    return

if not self.val.ValidarFecha(dia, mes, anio):
    messagebox.showerror("Error", "Fecha inválida / Invalid date")
    return

# Genera el RFC con formato simple / Generates RFC in simple format
rfc = ap_pat[0:2].upper() + ap_mat[0].upper() + nombre[0].upper() + anio[-2:] + mes.zfill(2) + dia.zfill(2)

# Agrega a pila o cola dependiendo del modo / Adds to stack or queue depending on mode
if self.modo.get() == "PILA":
    self.pila.append(rfc)
else:
    self.cola.append(rfc)

# Agrega al listbox / Adds to listbox
self.lista.insert(END, rfc)
messagebox.showinfo("Correcto", "RFC agregado correctamente / RFC added successfully")

# Limpia las cajas de texto / Clears text boxes
for caja in [self.nom, self.ap, self.am, self.dia, self.mes, self.anio]:
    caja.delete(0, END)

def eliminarElemento(self):
    # Elimina un elemento según el modo (pila o cola) / Removes an element depending on mode (stack or queue)
    if self.modo.get() == "PILA":
        if not self.pila:
            messagebox.showerror("Error", "Pila vacía / Stack is empty")
```



```
        return
    eliminado = self.pila.pop()
else:
    if not self.cola:
        messagebox.showerror("Error", "Cola vacía / Queue is empty")
        return
    eliminado = self.cola.popleft()

# Actualiza el listbox / Updates listbox
self.lista.delete(0, END)
for elem in (self.pila if self.modo.get() == "PILA" else list(self.cola)):
    self.lista.insert(END, elem)

messagebox.showinfo("Eliminado", f"Se eliminó: {eliminado} / Deleted: {eliminado}")

def inicio(self):
    # Etiquetas y cajas de texto / Labels and text boxes
    Label(self.ven, text="Nombre / Name").place(x=10, y=10)
    self.nom = Entry(self.ven, fg="blue")
    self.nom.place(x=10, y=35, width=150)

    Label(self.ven, text="Apellido paterno / Last name (father)").place(x=180, y=10)
    self.ap = Entry(self.ven, fg="blue")
    self.ap.place(x=180, y=35, width=150)

    Label(self.ven, text="Apellido materno / Last name (mother)").place(x=350, y=10)
    self.am = Entry(self.ven, fg="blue")
    self.am.place(x=350, y=35, width=150)

    # Campos de fecha / Date fields
    Label(self.ven, text="Día / Day").place(x=10, y=80)
    self.dia = Entry(self.ven, fg="green", width=5)
    self.dia.place(x=10, y=105, width=60)

    Label(self.ven, text="Mes / Month").place(x=90, y=80)
```



```
self.dia = Entry(self.ven, fg="green", width=5)
self.dia.place(x=10, y=105, width=60)

Label(self.ven, text="Mes / Month").place(x=90, y=80)
self.mes = Entry(self.ven, fg="green", width=5)
self.mes.place(x=90, y=105, width=60)

Label(self.ven, text="Año / Year").place(x=170, y=80)
self.anio = Entry(self.ven, fg="green", width=6)
self.anio.place(x=170, y=105, width=80)

# Botones principales / Main buttons
Button(self.ven, text="Agregar RFC / Add RFC", command=self.generarRFC).place(x=270, y=100, width=100, height=25)
Button(self.ven, text="Eliminar / Delete", command=self.eliminarElemento).place(x=390, y=100, width=100, height=25)

# Radio buttons para elegir modo / Radio buttons to choose mode
Label(self.ven, text="Modo de eliminación / Deletion mode:").place(x=10, y=150)
Radiobutton(self.ven, text="Pila (LIFO)", variable=self.modo, value="PILA").place(x=10, y=175)
Radiobutton(self.ven, text="Cola (FIFO)", variable=self.modo, value="COLA").place(x=120, y=175)

# Listbox para mostrar RFCs / Listbox to show RFCs
Label(self.ven, text="RFC Generados / Generated RFCs").place(x=10, y=210)
self.lista = Listbox(self.ven)
self.lista.place(x=10, y=235, width=520, height=130)

# Inicia la interfaz / Starts the interface
self.ven.mainloop()

# Ejecutar el programa principal / Run main program
if __name__ == "__main__":
    app = Principal()
    app.inicio()
```



## Programa 2 Validaciones

```

class Validar:
    def __init__(self):
        # Constructor vacío / Empty constructor
        pass

    def ValidarLetras(self, texto):
        # Verifica que el texto contenga solo letras o espacios
        # Checks that the text contains only letters or spaces
        if texto == "":
            return False
        for c in texto:
            if not ((ord(c) >= 65 and ord(c) <= 90) or (ord(c) >= 97 and ord(c) <= 122) or c == " "):
                return False
        return True

    def ValidarNumero(self, texto):
        # Verifica que el texto contenga solo números
        # Checks that the text contains only numbers
        if texto == "":
            return False
        for c in texto:
            if not (ord(c) >= 48 and ord(c) <= 57):
                return False
        return True

    def ValidarFecha(self, dia, mes, anio):
        # Verifica que los campos de fecha sean válidos
        # Checks that the date fields are valid
        if not (self.ValidarNumero(dia) and self.ValidarNumero(mes) and self.ValidarNumero(anio)):
            return False
        d, m, a = int(dia), int(mes), int(anio)
        # Valida los rangos de día, mes y año / Validates the ranges of day, month, and year
        if d < 1 or d > 31 or m < 1 or m > 12 or a < 1900 or a > 2025:
            return False
        return True

```

## Programa 3

Este programa crea una interfaz gráfica en Tkinter que permite registrar información personal: nombre, teléfono, domicilio y sexo (Femenino o Masculino).

Cada campo tiene un texto guía (placeholder) que desaparece al escribir.

Cuando el usuario llena todos los datos y presiona el botón “Agregar”, el programa valida que:

- El nombre contenga solo letras.
- El teléfono contenga solo números.
- El domicilio permita letras, números y espacios.



Si todo es correcto, genera una clave única formada por la primera letra de cada campo y muestra el registro completo en una lista (Listbox).

Si falta algún dato, aparece un mensaje de error.

Además, se incluye la opción de seleccionar el sexo mediante radiobuttons.

```
from tkinter import *
from tkinter import messagebox
from ValidacionP3 import Validar
import numpy as np

class Principal():
    def __init__(self):
        self.val = Validar() # Crea un objeto para validaciones / Create validation object
        self.ven = Tk()       # Crea la ventana principal / Create main window
        self.ven.title("Practica 3")

        # Configuración de tamaño y centrado / Window size and centering
        ancho = 350
        alto = 250
        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}")

# ----- FUNCIONES PARA PLACEHOLDER (texto guía en cajas) / Placeholder functions -----
def quitar_placeholder1(self,event):
    # Elimina el texto guía cuando el usuario hace clic / Remove placeholder when focused
    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):
    # Si está vacío, vuelve a poner el texto guía / If empty, restore placeholder
    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")

# ----- INTERFAZ PRINCIPAL / MAIN INTERFACE -----
def inicio(self):
    # Caja de texto: Nombre / Entry box: Name
    self.placeholder1 = "Nombre"
    self.nombre = Entry(self.ven, fg="gray")
    self.nombre.insert(0, self.placeholder1)
    self.nombre.bind("<FocusIn>", self.quitar_placeholder1)
    self.nombre.bind("<FocusOut>", self.poner_placeholder1)
    #self.nombre.bind("<Return>", self.validarCaja)
    self.nombre.place(x=10, y=10, width=100)

    # Caja de texto: Teléfono / Entry box: Phone
    self.placeholder2 = "Telefno"
    self.telefono = Entry(self.ven, fg="gray")
    self.telefono.insert(0, self.placeholder2)
    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=10, width=100)
```



```
# Caja de texto: Domicilio / Entry box: Address
self.placeholder3 = "Domicilio"
self.domicilio = Entry(self.ven, fg="gray")
self.domicilio.insert(0, self.placeholder3)
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)

# Etiqueta para el sexo / Label for gender
Label(self.ven, text="Sexo").place(x=10, y=30)

# Radio buttons para seleccionar sexo / Radio buttons for gender selection
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)

# Listbox donde se muestran los registros / Listbox to show records
self.lista= Listbox(self.ven, height=6, width=40, bg="white", font=("Helvetica",12))
self.lista.place(x=10, y=100)

# Botón para agregar registro / Button to add record
Button(self.ven, text="Agregar", command=self.validarCaja, width=10).place(x=100, y=50, height=50, width=100)

# Mantener ventana activa / Keep window open
self.ven.mainloop()

# ----- VALIDACIÓN Y REGISTRO / VALIDATION AND RECORD INSERTION -----
def validarCaja(self,event=0):
    # Verifica si faltan datos / Check if any field is empty
    if (self.nombre.get() == self.placeholder1 or self.telefono.get() == self.placeholder2
        or self.domicilio == self.placeholder3):
        messagebox.showinfo("Error", "Faltan datos / Missing data")
    else:
        # Obtiene los valores de las cajas / Get entry values
```



```
# Obtiene los valores de las cajas / Get entry values
nombre = self.nombre.get()
telefono = self.telefono.get()
domicilio = self.domicilio.get()

# Determina el sexo seleccionado / Determine selected gender
if self.modo.get() == "F":
    Sexo = "Femenino"
else:
    Sexo = "Masculino"

# Crea una clave única a partir de las iniciales / Create unique key from initials
Clave = nombre[0] + telefono[0] + domicilio[0]

# Combina todos los datos en un registro / Combine data into one record
Persona = Clave + "-" + nombre + "-" + telefono + "-" + domicilio + "-" + Sexo

# Agrega el registro al Listbox / Add record to listbox
self.lista.insert(self.lista.size()+1, Persona)

# ----- PROGRAMA PRINCIPAL / MAIN PROGRAM -----
if __name__ == '__main__':
    app = Principal()
    app.inicio()

# validar para que nombre no numeros, telefono no letras
# validate so name cannot contain numbers and phone cannot contain letters
```



```
class Validar:
    def __init__(self):
        # Contador para validaciones (por si se necesita en el futuro)
        # Counter for validations (reserved for future use)
        self.con = 0

    def ValidarNumeros(self, num):
        # Verifica que todos los caracteres sean números
        # Checks that all characters are numbers
        if num == "":
            return False # Si está vacío / If empty, return False
        for c in num:
            # Comprueba si el código ASCII está entre 48 ('0') y 57 ('9')
            # Checks if ASCII code is between 48 ('0') and 57 ('9')
            if not (ord(c) >= 48 and ord(c) <= 57):
                return False
        return True # Si todos son números / If all are digits

    def ValidarLetras(self, texto):
        # Verifica que todos los caracteres sean letras o espacios
        # Checks that all characters are letters or spaces
        if texto == "":
            return False # Si está vacío / If empty, return False
        for c in texto:
            # Comprueba si es una letra mayúscula, minúscula o espacio
            # Checks if uppercase, lowercase, or space
            if not ((ord(c) >= 65 and ord(c) <= 90) or
                    (ord(c) >= 97 and ord(c) <= 122) or
                    c == " "):
                return False
        return True # Si todos los caracteres son válidos / If all characters valid
```



```
def ValidarDomicilio(self, texto):
    # Verifica que el domicilio contenga letras, números o espacios
    # Checks that address contains letters, numbers, or spaces
    if texto == "":
        return False # No se permite vacío / Empty not allowed
    for c in texto:
        # Comprueba si es letra, número o espacio
        # Checks if character is letter, number, or space
        if not ((ord(c) >= 65 and ord(c) <= 90) or
                (ord(c) >= 97 and ord(c) <= 122) or
                (ord(c) >= 48 and ord(c) <= 57) or
                c == " "):
            return False
    return True # Si el domicilio es válido / If address is valid
```

### Validaciones Programa 3

Practica 3

Nombre: Leslie | Número de Identidad: 3441035639 | Domicilio: Hidalgo 2005

Sexo:

F  M

Agregar

L3H-Leslie-3441035639-Hidalgo 2005-Femenina



## Programa 4

Crea una ventana con campos de entrada: nombre, edad y correo electrónico.

Permite agregar, seleccionar, modificar y eliminar registros en una tabla (Treeview).

Usa la clase Validar para asegurar que el nombre tenga solo letras, la edad solo números y el correo incluya '@'.

```
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
from valP4 import Validar
import numpy as np
import random

class Principal():
    def __init__(self):
        # Se crea el objeto de validación / Create validation object
        self.val = Validar()
        # Se crea la ventana principal / Create main window
        self.ven = Tk()
        self.ven.title('Practica 4')

        # Configuración de tamaño y posición de la ventana / Window size and position
        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}")

        # Variables de control / Control variables
        self.cont = 0          # Contador para generar claves únicas / Counter for unique keys
        self.bandera = False   # Indica si se está editando un registro / Indicates edit mode
        self.renglon = -1      # Guarda el índice del renglón seleccionado / Selected row index
        self.index = ""         # Clave base para edición / Base key for editing

    def validarCaja(self):
        # Obtiene el renglón seleccionado en la tabla / Get selected row in table
        self.renglon = self.tabla.selection()
        if not self.renglon:
            messagebox.showerror("Error","Elige una fila") # Mensaje si no hay selección / No selection message
        else:
            # Obtiene los valores de la fila / Get row values
```



```
valores = self.tabla.item(self.renglon, "values")
print(valores)
# Extrae el índice base de la clave / Extract base index from key
self.index = valores[0]
self.index = self.index[:len(self.index)-2]
print(self.index)
# Carga los valores en las cajas de texto / Load values into textboxes
self.nombre.insert(0,valores[1])
self.edad.insert(0,valores[3])
self.correo.insert(0,valores[2])
# Activa modo edición / Enable edit mode
self.bandera= True

def agregarElemento(self):
    # Validación de campos vacíos / Check for empty fields
    if(len(self.nombre.get())==0 or len(self.edad.get())==0 or len(self.correo.get())==0):
        messagebox.showerror("Error","Faltan datos")
    else:
        # Validaciones de nombre, edad y correo / Validate name, age, and email
        if (self.val.ValidarNombre(self.nombre.get()) and
            self.val.ValidarEdad(self.edad.get()) and
            self.val.ValidarCorreo(self.correo.get())):
            # Obtiene los valores / Get values
            nombre = self.nombre.get()
            edad = self.edad.get()
            correo = self.correo.get()

            # Si no está en modo edición / If not in edit mode
            if self.bandera == False:
                self.cont += 1
                # Genera una clave única / Generate unique key
                clave = str(self.cont)+str(random.randint(1,100))+self.nombre.get()[0:2].upper()
                # Inserta los datos en la tabla / Insert data into table
                self.tabla.insert("", "end", values=(clave,nombre,correo,edad))
```



```
# Limpia las cajas de texto / Clear textboxes
self.nombre.delete(0,END)
self.edad.delete(0,END)
self.correo.delete(0,END)
else:
    # Modo edición activado / Edit mode active
    clave = self.index+self.nombre.get()[0:2].upper()
    print("Modo edición activado")
    # Actualiza los datos en la tabla / Update data in table
    self.tabla.item(self.renglon, values=(clave,nombre,correo,edad))
    # Limpia campos y restablece variables / Clear inputs and reset flags
    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")
else:
    # Mensaje de error en validación / Validation error message
    messagebox.showinfo("Incorrecto",
        "Verifica los datos:\n- Nombre: solo letras\n- Edad: solo números\n- Correo: debe contener '@'")

def eliminar(self):
    # Obtiene la fila seleccionada / Get selected row
    renglon = self.tabla.selection()
    if not renglon:
        messagebox.showerror("Error","Elige una fila")
    else:
        # Elimina la fila seleccionada / Delete selected row
        self.tabla.delete(renglon)
        messagebox.showinfo("Correcto","Fila eliminada")

def inicio(self):
    # Etiquetas y cajas de texto / Labels and input fields
```



```
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)

# Botones de acción / Action buttons
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="Seleccionar", command=self.validarCaja, width=10).place(x=380,y=130, width=100,height=30)

# Tabla para mostrar datos / Table to display data
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)

# Configura encabezados de tabla / Configure table headers
for col in columnas:
    self.tabla.heading(col,text=col)
    self.tabla.column(col, anchor="center", width=30)

# Barras de desplazamiento / Scrollbars
scrollly = ttk.Scrollbar(self.ven,orient="vertical", command=self.tabla.yview)
scrolllx = ttk.Scrollbar(self.ven, orient="horizontal", command=self.tabla.xview)
scrollly.place(x=360,y=90,height=200)
scrolllx.place(x=10,y=280, width=350)
```

```
        self.ven.mainloop()
```

```
# Programa principal / Main program
if __name__=='__main__':
    app = Principal()
    app.inicio()
```

