

**UNIVERSIDAD MAYOR DE SAN ANDRÉS  
FACULTAD DE TECNOLOGÍA  
ELECTRÓNICA Y TELECOMUNICACIONES**



***PRACTICA N°1  
INFORMATICA SUPERIOR I***

**Estudiante:** Univ. Surco Nina Williams Rodrigo  
**CI:** 7022363 LP.  
**RU:** 1877364  
**Docente:** Lic. Julia Torrez Soria  
**Semestre:** I/2025

LA PAZ – BOLIVIA  
2025

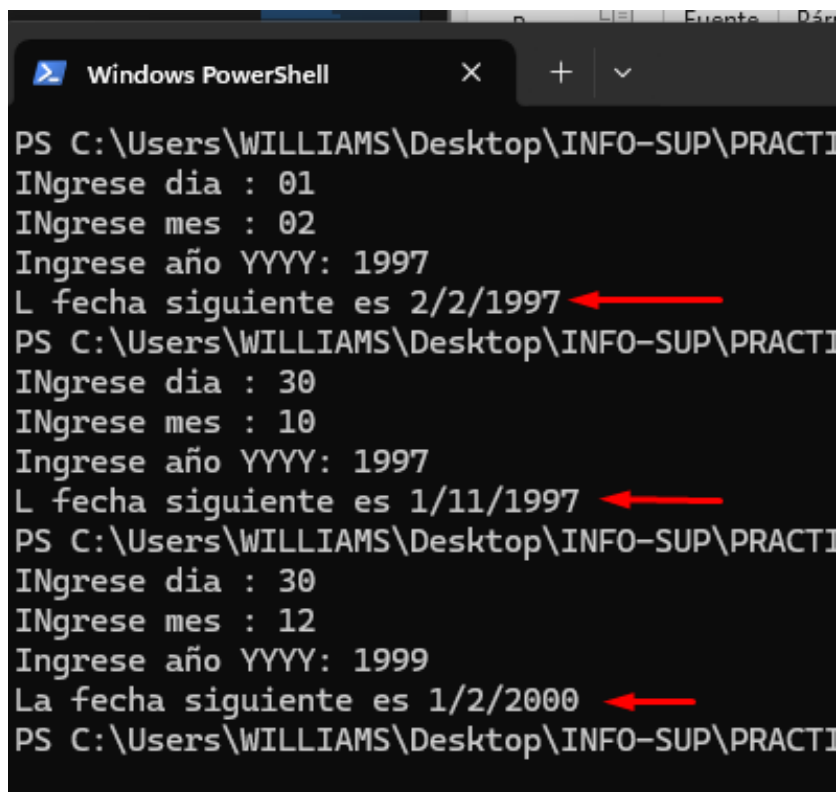
1. Realizar un programa que permita pedir el día, mes y año de una fecha correcta y mostrar la fecha del día siguiente. suponer que todos los meses tienen 30 días.

### CÓDIGO

```
dia = int(input("Ingrese día : "))
mes = int(input("Ingrese mes : "))
año = int(input("Ingrese año YYYY: "))

if dia < 30:
    print(f"L fecha siguiente es {dia + 1}/{mes}/{año}")
else:
    dia = 1
    if mes < 12:
        print(f"L fecha siguiente es {dia}/{mes + 1}/{año}")
    else:
        mes = 1
        print(f"La fecha siguiente es {dia}/{mes + 1}/{año + 1}")
```

### CAPTURA



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTI
Ingrese día : 01
Ingrese mes : 02
Ingrese año YYYY: 1997
L fecha siguiente es 2/2/1997
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTI
Ingrese día : 30
Ingrese mes : 10
Ingrese año YYYY: 1997
L fecha siguiente es 1/11/1997
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTI
Ingrese día : 30
Ingrese mes : 12
Ingrese año YYYY: 1999
La fecha siguiente es 1/2/2000
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTI
```

2. Realizar un programa que determine el menor de entre grupo de números enteros.

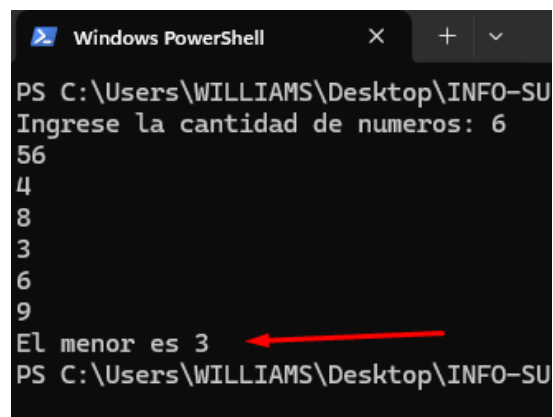
### CÓDIGO

```
menor = 10**100
n = int(input("Ingrese la cantidad de numeros: "))

for _ in range(n):
    valor = int(input())
    if valor < menor:
        menor = valor

print(f"El menor es {menor}")
```

### CAPTURA



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SU
Ingrese la cantidad de numeros: 6
56
4
8
3
6
9
El menor es 3
PS C:\Users\WILLIAMS\Desktop\INFO-SU
```

3. Realizar un programa que reciba 3 números enteros y luego:
  - a) Muestre el número intermedio del conjunto de los 3 números.
  - b) Presente los 3 números en forma descendente.

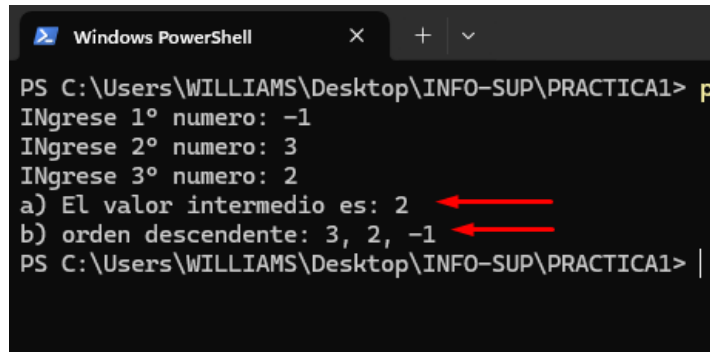
### CÓDIGO

```
a = int(input("INGrese 1° numero: "))
b = int(input("INGrese 2° numero: "))
c = int(input("INGrese 3° numero: "))
intermedio = a + b + c - min(a, b, c) - max(a, b, c)

print(f"a) El valor intermedio es: {intermedio}")

print(f"b) orden descendente: {max(a,b,c)}, {intermedio}, {min(a,b,c)}")
```

### CAPTURA



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> p
Ingrese 1º numero: -1
Ingrese 2º numero: 3
Ingrese 3º numero: 2
a) El valor intermedio es: 2
b) orden descendente: 3, 2, -1
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> |
```

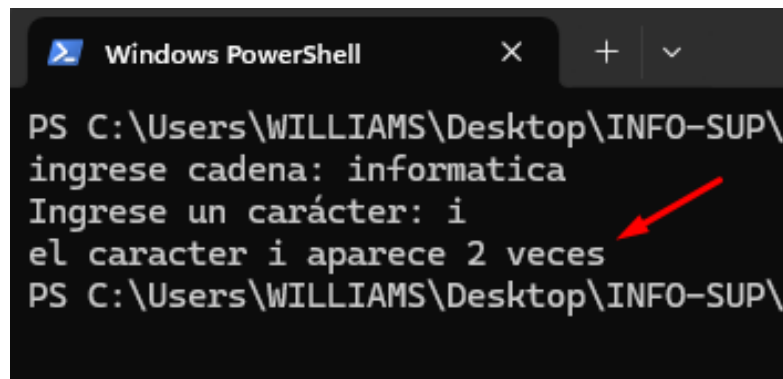
4. Realizar un programa que pida una cadena y un carácter por teclado (valida que sea un carácter) y muestra cuantas veces aparece el carácter en la cadena.

### CÓDIGO

```
cadena = input("ingrese cadena: ")
char = ""
while True:
    char = input("Ingresa un carácter: ")
    if len(char) == 1 and char.isprintable():
        break
    print("Entrada inválida. Debe ingresar un solo carácter.")
c = 0
for i in range(len(cadena)):
    if cadena[i] == char:
        c += 1

print(f"el caracter {char} aparece {c} veces")
```

### CAPTURA



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\
ingrese cadena: informatica
Ingresa un carácter: i
el caracter i aparece 2 veces
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\
```

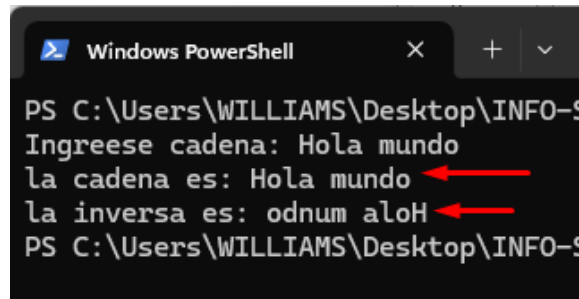
5. Realizar un programa que permita ingresar por teclado una cadena la misma que debe ser invertida, mostrar por pantalla ambas cadenas.

## CÓDIGO

```
cad = input("Ingreese cadena: ")

print(f"la cadena es: {cad}")
print(f"la inversa es: {cad[::-1]}")
```

## CAPTURA



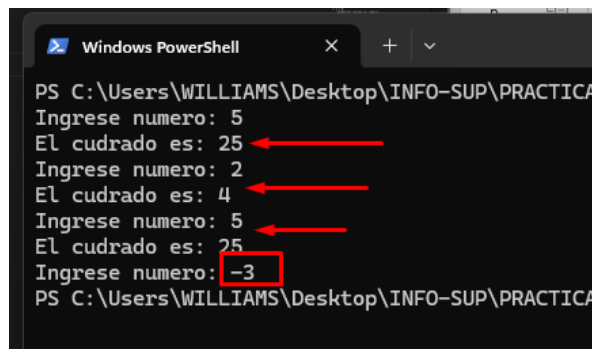
```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\
Ingreese cadena: Hola mundo
la cadena es: Hola mundo
la inversa es: odnum aloH
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\
```

- Realizar un programa que permita leer un número y mostrar su cuadrado, repetir el proceso hasta que se introduzca un número negativo.

## CÓDIGO

```
while True:
    n = int(input("Ingreese numero: "))
    if n > 0:
        print(f"El cudrado es: {n*n}")
    else:
        break
```

## CAPTURA



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA
Ingreese numero: 5
El cudrado es: 25
Ingreese numero: 2
El cudrado es: 4
Ingreese numero: 5
El cudrado es: 25
Ingreese numero: -3
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA
```

- Realizar la operación de potenciación ( $a^b$ ), de dos valores enteros positivos, con multiplicaciones.

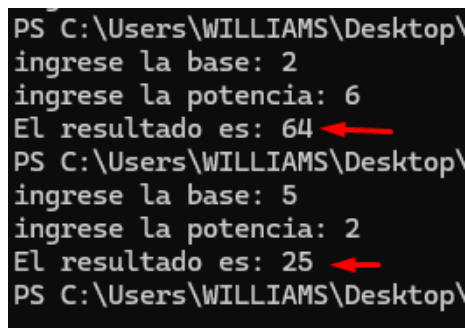
## CÓDIGO

```
a = int(input("ingrese la base: "))
b = int(input("ingrese la potencia: "))

sum = 1
for _ in range (b):
    sum *= a

print(f"El resultado es: {sum}")
```

## CAPTURA



```
PS C:\Users\WILLIAMS\Desktop\
ingrese la base: 2
ingrese la potencia: 6
El resultado es: 64
PS C:\Users\WILLIAMS\Desktop\
ingrese la base: 5
ingrese la potencia: 2
El resultado es: 25
PS C:\Users\WILLIAMS\Desktop\
```

8. Escribir un programa que sume las cifras de un número entero positivo.

Ejemplo:             $962 \rightarrow 9 + 6 + 2 = 17$   
                      $17 \rightarrow 1 + 7 = 8$   
                     El resultado es 8

## CÓDIGO

```
n = int(input("Ingresar numero:"))
suma = 0
if n > 9:
    while n > 9:
        d = n % 10
        n = n // 10
        suma += d
    if n < 10:
        suma += n
        n = suma
        suma = 0
print(f"la suma es: {n}")
```

## CAPTURA

```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SU
Ingresar numero:962
La suma es: 8
PS C:\Users\WILLIAMS\Desktop\INFO-SU
Ingresar numero:111111111
La suma es: 1
PS C:\Users\WILLIAMS\Desktop\INFO-SU
Ingresar numero:9
La suma es: 9
PS C:\Users\WILLIAMS\Desktop\INFO-SU
```

9. Realizar un programa que invierta un número introducido por teclado. Debe solicitar un valor entero y mostrar el mismo número con sus cifras invertidas. Si el número es negativo debe seguir siéndolo.

|        -12345 → -54321  
          54300 → 00345

#### CÓDIGO

```
n = input("INGrese numero: ")

if n[0] == "-":
    print("-"+n[:0:-1])
else:
    print(n[::-1])
```

#### CAPTURA

```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\I
INGrese numero: -12345
-54321
PS C:\Users\WILLIAMS\Desktop\I
INGrese numero: 54300
00345
PS C:\Users\WILLIAMS\Desktop\I
```

10. Para un conjunto de n datos reales se desea determinar el mayor de los datos negativos y cuantas veces aparece.

Por ejemplo, para n=6, datos: 4,-5,6,-2,5,-3.

## CÓDIGO

```
n = int(input("Ingrese la cantidad DE NUMEROS: "))

vector = []

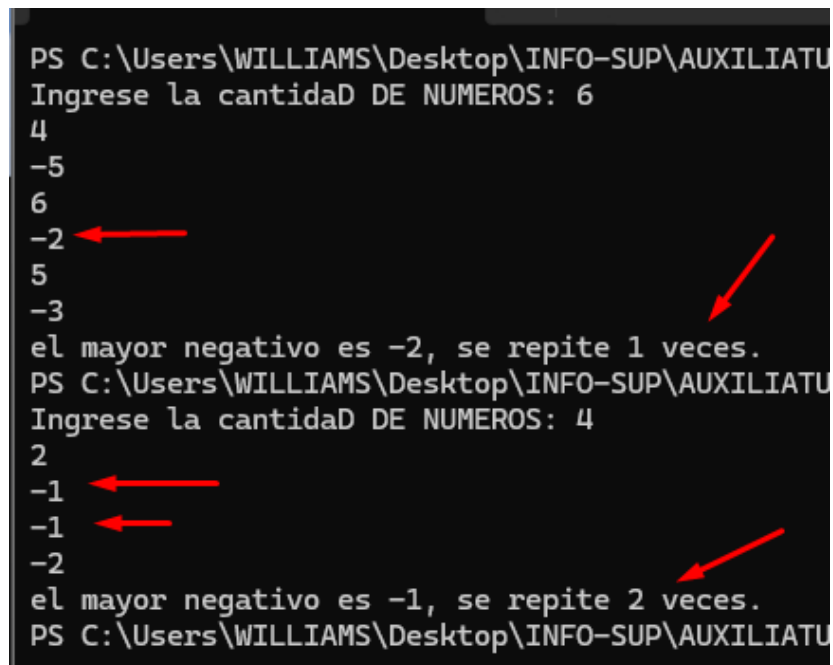
for _ in range(n):
    valor = int(input())
    if valor < 0:
        vector.append(valor)

vector.sort()

contador = 0

for i in vector:
    if (i - vector[len(vector)-1]) == 0:
        contador += 1
if len(vector) != 0:
    print(f"el mayor negativo es {vector[len(vector)-1]}, se repite {contador} veces.")
else:
    print("No hay negativos")
```

## CAPTURA



```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATU
Ingrese la cantidad DE NUMEROS: 6
4
-5
6
-2
5
-3
el mayor negativo es -2, se repite 1 veces.
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATU
Ingrese la cantidad DE NUMEROS: 4
2
-1
-1
-2
el mayor negativo es -1, se repite 2 veces.
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATU
```

11. Escribir un programa que muestre la figura de caracteres siguiente. El valor del carácter máximo mostrado se le debe pedir al usuario. Los únicos valores válidos son del 'a' a la 'i'.



```

      a
    a b a
  a b c b a
a b c d c b a
  a b c b a
    a b a
      a

```

## CÓDIGO

```

while True:
    char = input("Ingrese un carácter entre 'a' e 'i': ")
    if len(char) == 1 and 'a' <= char <= 'i':
        break
    print("Invalido. Debe ingresar un solo carácter entre 'a' e 'i'.")

max_char = ord(char)

#Superior
for i in range(ord('a'), max_char + 1):
    espacios = max_char - i
    print(" " * espacios, end="")
    for j in range(ord('a'), i + 1):
        print(chr(j), end=" ")
    for j in range(i - 1, ord('a') - 1, -1):
        print(chr(j), end=" ")
    print()

# Inferior
for i in range(max_char - 1, ord('a') - 1, -1):
    espacios = max_char - i
    print(" " * espacios, end="")
    for j in range(ord('a'), i + 1):
        print(chr(j), end=" ")
    for j in range(i - 1, ord('a') - 1, -1):
        print(chr(j), end=" ")
    print()

```

## CAPTURA

```

Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRA
Ingrese un carácter entre 'a' e 'i': d
  a
 a b a
a b c b a
a b c d c b a
 a b c b a
  a b a
    a
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRA
Ingrese un carácter entre 'a' e 'i': f
  a
 a b a
 a b c b a
a b c d c b a
a b c d e d c b a
a b c d e f e d c b a
 a b c d e d c b a
  a b c d c b a
    a b c b a
      a b a
        a
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRA

```

12. El programa debe dar un mensaje de error cuando el número N no está en los límites indicados y solicitar nuevamente el valor N al usuario.

Ejemplo para N=4

```

- + + +
* - + +
* * - +
* * * -

```

## CÓDIGO

```

n = int(input("Ingrese N: "))

for i in range(n):
    for j in range(n):
        if j == i:
            print("-", end= " ")
        elif i > j:
            print("*", end = " ")
        else:
            print("+", end = " ")
    print()

```

## CAPTURA

```
Windows PowerShell
PS C:\Users\WILLIAMS\I...
Ingrese N: 4
- + + +
* - + +
* * - +
* * * -
PS C:\Users\WILLIAMS\I...
Ingrese N: 6
- + + + + +
* - + + + +
* * - + + +
* * * - + +
* * * * - +
* * * * * -
PS C:\Users\WILLIAMS\I...
```

13. Los pacientes con síntomas de una cierta enfermedad son ingresados en el hospital si tienen un valor superior a 0.6 en la medición de un determinado índice, y son operados si el valor es superior a 0.9. Escribir un programa que lea desde teclado el número de pacientes seguido de la edad y el índice de cada paciente, y calcule la edad media de los pacientes analizados, así como la edad media de los ingresados y la edad media de los operados.

### CÓDIGO

```
# Leer la edad y el índice de cada paciente
for _ in range(num_pacientes):
    edad = int(input("Ingrese la edad del paciente: "))
    indice = float(input("Ingrese el índice del paciente: "))

    total_edad += edad

    if indice > 0.6:
        num_ingresados += 1
        edad_ingresados += edad

    if indice > 0.9:
        num_operados += 1
        edad_operados += edad

# Calcular y mostrar las edades medias
print("Edad media de los pacientes analizados:", total_edad / num_pacientes)
print("Edad media de los ingresados:", (edad_ingresados / num_ingresados) if num_ingresados > 0 else "No existe")
print("Edad media de los operados:", (edad_operados / num_operados) if num_operados > 0 else "No existe")
```

### CAPTURA

```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> py .\ej
Ingrese el número de pacientes: 3
Ingrese la edad del paciente: 30
Ingrese el índice del paciente: 0.5
Ingrese la edad del paciente: 40
Ingrese el índice del paciente: 0.7
Ingrese la edad del paciente: 50
Ingrese el índice del paciente: 0.95
Edad media de los pacientes analizados: 40.0
Edad media de los ingresados: 45.0
Edad media de los operados: 50.0
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> |
```

14. Realizar un programa que pida números y los guarda en una lista, cuando el usuario meta un 0 ya dejaremos de insertar. Por último, muestra los números ordenados de menor a mayor.

### CÓDIGO

```
lista = []

while True:
    n = int(input("Ingrese num: "))

    if n == 0:
        break

    lista.append(n)

n = len(lista)
for i in range(n):
    for j in range(0, n - i - 1):
        if lista[j] > lista[j + 1]:
            lista[j], lista[j + 1] = lista[j + 1], lista[j]
print("ordenada:", lista)
```

### CAPTURA

```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> py .
Ingrese num: 6
Ingrese num: 7
Ingrese num: 56
Ingrese num: 7
Ingrese num: 23
Ingrese num: 2
Ingrese num: 4
Ingrese num: 1
Ingrese num: 0
ordenada: [1, 2, 4, 6, 7, 7, 23, 56]
```

15. Realizar un programa que tome un string y devuelva un diccionario cuyas llaves son las letras del string y sus respectivos valores son la cantidad de veces que ellas aparecen en el string. Por Ejemplo, contador('telecomunicaciones') {'t': 1, 'e': 3, 'l': 1, ...}..

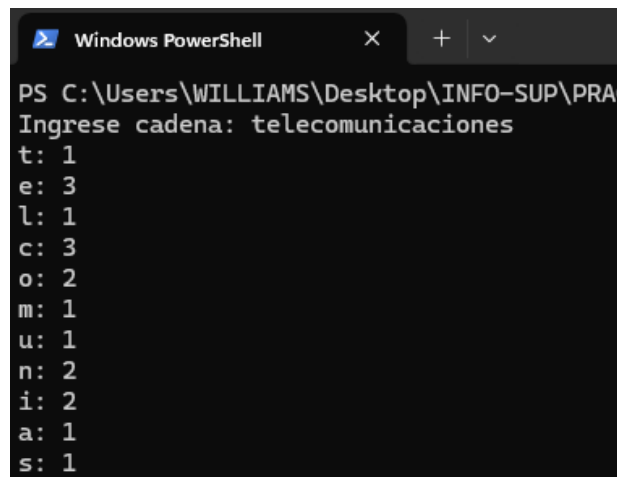
#### CÓDIGO

```
cad = input("Ingrese cadena: ")
dic = {}

for i in range(len(cad)):
    if cad[i] in dic:
        dic[cad[i]] = dic[cad[i]] + 1
    else:
        dic[cad[i]] = 1

for i in dic.items():
    print(f"{i[0]}: {i[1]}")
```

#### CAPTURA



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRA
Ingrese cadena: telecomunicaciones
t: 1
e: 3
l: 1
c: 3
o: 2
m: 1
u: 1
n: 2
i: 2
a: 1
s: 1
```

16. Realizar un programa que intercambie claves y valores en un diccionario.

#### CÓDIGO

```
def intercambiar_claves_valores(diccionario):
    nuevo_diccionario = {}

    for clave, valor in diccionario.items():
        nuevo_diccionario[valor] = clave

    return nuevo_diccionario

diccionario_original = {
    "nombre": "Carlos",
    "edad": 25,
```

```

        "ciudad": "La Paz"
    }

    diccionario_nuevo = intercambiar_claves_valores(diccionario_original)

    # Mostrar los resultados
    print("Diccionario original:")
    for clave, valor in diccionario_original.items():
        print(f"{clave}: {valor}")
    print()
    print("\nDiccionario intercambiado:")
    for clave, valor in diccionario_nuevo.items():
        print(f"{clave}: {valor}")

```

## CAPTURA

```

PS C:\Users\WILLIAMS\Desktop\I
Diccionario original:
nombre: Carlos
edad: 25
ciudad: La Paz

Diccionario intercambiado: ➡
Carlos: nombre
25: edad
La Paz: ciudad
PS C:\Users\WILLIAMS\Desktop\I

```

17. Realizar un programa aplicando la POO de la clase Artículo científico:

| ArticuloCientifico   |
|--|
| Titulo:String<br>Autor:String<br>PalabrasClaves:String<br>Publicación:String<br>Año:int<br>Resumen: String |
| __init__()<br>Mostrar_articulos()  |

a) Instanciar N objetos y almacenarlos en una lista.

## CÓDIGO

```
class ArtículoCientifico:
    def __init__(self, titulo, autor, palabras_clave, publicacion, anio, resumen):
        self.titulo = titulo
        self.autor = autor
        self.palabras_clave = palabras_clave
        self.publicacion = publicacion
        self.anio = anio
        self.resumen = resumen

    def mostrar_articulo(self):
        print(f"Título: {self.titulo}")
        print(f"Autor: {self.autor}")
        print(f"Palabras Clave: {self.palabras_clave}")
        print(f"Publicación: {self.publicacion}")
        print(f"Año: {self.anio}")
        print(f"Resumen: {self.resumen}")
        print("-" * 40)

# Lista para almacenar los artículos
articulos = []

# Crear e instanciar N objetos
N = int(input("Ingrese la cantidad de artículos científicos: "))
for i in range(N):
    print(f"\nRegistro del artículo {i+1}:")
    titulo = input("Título: ")
    autor = input("Autor: ")
    palabras_clave = input("Palabras Clave: ")
    publicacion = input("Publicación: ")
    anio = int(input("Año: "))
    resumen = input("Resumen: ")

    articulo = ArtículoCientifico(titulo, autor, palabras_clave, publicacion, anio, resumen)
    articulos.append(articulo)

# Mostrar todos los artículos almacenados
print("\nLista de artículos científicos registrados:\n")
for articulo in articulos:
    articulo.mostrar_articulo()
```

## CAPTURA

```

Ingrese la cantidad de artículos científicos: 3

Registro del artículo 1:
Título: Fabulas
Autor: Ruben Gomez
Palabras Clave: cuento
Publicación: 2015
Año: 2022
Resumen: trata de cuentyos varios

Registro del artículo 2:
Título: algebra
Autor: Sebastian lazo
Palabras Clave: matematica
Publicación: 2015
Año: 2013
Resumen: ejercicios matematicos

Registro del artículo 3:
Título: Calculo 1
Autor: Chungara
Palabras Clave: calculo
Publicación: 2000
Año: 1999
Resumen: teoria calculo de una variable

```

```

Lista de artículos científicos registrados:

```

```

Título: Fabulas
Autor: Ruben Gomez
Palabras Clave: cuento
Publicación: 2015
Año: 2022
Resumen: trata de cuentyos varios
-----
Título: algebra
Autor: Sebastian lazo
Palabras Clave: matematica
Publicación: 2015
Año: 2013
Resumen: ejercicios matematicos
-----
Título: Calculo 1
Autor: Chungara
Palabras Clave: calculo
Publicación: 2000
Año: 1999
Resumen: teoria calculo de una variable
-----
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> |

```

18. Realizar un programa aplicando la POO de la clase Agenda Telefonica:

| AgendaTelefonica   |
|--|
| Nombre:String<br>Apellido:String<br>NroCelular:String<br>Correo:String       |
| __init__()<br>Adicionar_Contacto()<br>Mostrar_Contactos()<br>Buscar_Contacto |



a) Instanciar N objetos y almacenarlos en una lista.

## CÓDIGO

```
class Contacto:
    def __init__(self, nombre, apellido, nro_celular, correo):
        self.nombre = nombre
        self.apellido = apellido
        self.nro_celular = nro_celular
        self.correo = correo

    def mostrar_info(self):
        print(f"Nombre: {self.nombre} {self.apellido}")
        print(f"Celular: {self.nro_celular}")
        print(f"Correo: {self.correo}")
        print("-" * 40)

class AgendaTelefonica:
    def __init__(self):
        self.contactos = [] # Lista para almacenar los contactos

    def adicionar_contacto(self, contacto):
        self.contactos.append(contacto)
        print(f"Contacto {contacto.nombre} {contacto.apellido} agregado con éxito.\n")

    def mostrar_contactos(self):
        if not self.contactos:
            print("La agenda está vacía.\n")
        else:
            print("\nLista de contactos:")
            for contacto in self.contactos:
                contacto.mostrar_info()

    def buscar_contacto(self, nombre):
        print(f"\nBuscando contacto con nombre: {nombre}...")
        encontrados = [c for c in self.contactos if c.nombre.lower() == nombre.lower()]

        if encontrados:
            print("\nContacto(s) encontrado(s):")
            for contacto in encontrados:
                contacto.mostrar_info()
        else:
            print("No se encontró ningún contacto con ese nombre.\n")

# Crear una agenda
```

```

agenda = AgendaTelefonica()

# Solicitar la cantidad de contactos a agregar
N = int(input("Ingrese la cantidad de contactos a registrar: "))

# Adicionar contactos a la agenda
for i in range(N):
    print(f"\nRegistro del contacto {i+1}:")
    nombre = input("Nombre: ")
    apellido = input("Apellido: ")
    nro_celular = input("Número de celular: ")
    correo = input("Correo: ")

    nuevo_contacto = Contacto(nombre, apellido, nro_celular, correo)
    agenda.adicionar_contacto(nuevo_contacto)

# Mostrar todos los contactos registrados
agenda.mostrar_contactos()
print("-"*30)

# Buscar un contacto por nombre
nombre_buscar = input("\nIngrese el nombre del contacto a buscar: ")
agenda.buscar_contacto(nombre_buscar)

```

## CAPTURA

```

PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> py .\ejercicio18.py
Ingrese la cantidad de contactos a registrar: 3

Registro del contacto 1:
Nombre: Williams
Apellido: Surco
Número de celular: 65550487
Correo: w@gmail.com
Contacto Williams Surco agregado con éxito.

Registro del contacto 2:
Nombre: Rene
Apellido: Nina
Número de celular: 79560076
Correo: m@gmail.com
Contacto Rene Nina agregado con éxito.

Registro del contacto 3:
Nombre: Marco alcoreza
Apellido: Alcoreza
Número de celular: 79157678
Correo: mm@gmail.com
Contacto Marco alcoreza Alcoreza agregado con éxito.

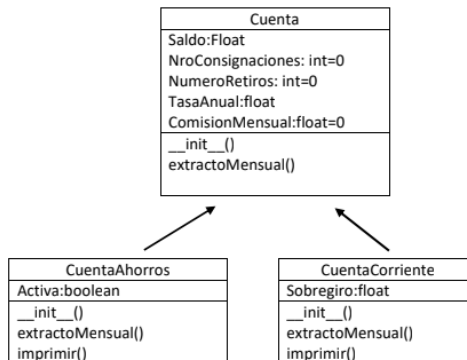
```

## Impresión de datos y búsqueda de un contacto.

```
Lista de contactos:
Nombre: Williams Surco
Celular: 65550487
Correo: w@gmail.com
-----
Nombre: Rene Nina
Celular: 79560076
Correo: m@gmail.com
-----
Nombre: Marco alcoreza Alcoreza
Celular: 79157678
Correo: mm@gmail.com
-----

Ingrese el nombre del contacto a buscar: Rene
Buscando contacto con nombre: Rene...
Contacto(s) encontrado(s):
Nombre: Rene Nina
Celular: 79560076
Correo: m@gmail.com
-----
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> |
```

19. Desarrollar un programa aplicando la POO de la clase padre Cuenta y las clases hijas Cuenta\_Corriente y Cuenta Ahorros.



- a) Instanciar 3 objetos
- b) Aplicar el principio de Herencia

## CÓDIGO

```
# Clase base Cuenta
class Cuenta:
    def __init__(self, saldo=0.0, tasa_anual=0.0):
        self.saldo = saldo
        self.nro_consignaciones = 0
        self.nro_retiros = 0
        self.tasa_anual = tasa_anual
        self.comision_mensual = 0.0

    def consignar(self, monto):
        if monto > 0:
```

```

        self.saldo += monto
        self.nro_consignaciones += 1
        print(f"Consignación exitosa. Nuevo saldo: {self.saldo}")
    else:
        print("El monto debe ser mayor a cero.")

def retirar(self, monto):
    if 0 < monto <= self.saldo:
        self.saldo -= monto
        self.nro_retiros += 1
        print(f"Retiro exitoso. Nuevo saldo: {self.saldo}")
    else:
        print("Saldo insuficiente o monto inválido.")

def extracto_mensual(self):
    self.saldo -= self.comision_mensual
    print(f"Extracto mensual aplicado. Saldo final: {self.saldo}")

def imprimir(self):
    print(f"\nSaldo: {self.saldo}")
    print(f"Número de consignaciones: {self.nro_consignaciones}")
    print(f"Número de retiros: {self.nro_retiros}")
    print(f"Tasa anual: {self.tasa_anual}%")
    print(f"Comisión mensual: {self.comision_mensual}\n")

# Clase hija CuentaAhorros
class CuentaAhorros(Cuenta):
    def __init__(self, saldo=0.0, tasa_anual=0.0, activa=False):
        super().__init__(saldo, tasa_anual)
        self.activa = activa if saldo > 0 else False

    def extracto_mensual(self):
        if self.nro_retiros > 4: # Se cobra una comisión si hay más de 4 retiros
            self.comision_mensual += 5.0
        super().extracto_mensual()

    def imprimir(self):
        super().imprimir()
        print(f"Cuenta Activa: {'Sí' if self.activa else 'No'}\n")

# Clase hija CuentaCorriente
class CuentaCorriente(Cuenta):
    def __init__(self, saldo=0.0, tasa_anual=0.0, sobregiro=0.0):

```

```

        super().__init__(saldo, tasa_anual)
        self.sobregiro = sobregiro

    def retirar(self, monto):
        if monto > self.saldo:
            self.sobregiro += (monto - self.saldo)
            self.saldo = 0
        else:
            super().retirar(monto)

    def extracto_mensual(self):
        if self.sobregiro > 0:
            self.saldo -= self.sobregiro
            self.sobregiro = 0
        super().extracto_mensual()

    def imprimir(self):
        super().imprimir()
        print(f"Saldo de sobregiro: {self.sobregiro}\n")

# Instanciación de objetos
cuenta1 = CuentaAhorros(saldo=500, tasa_anual=3.5, activa=True)
cuenta2 = CuentaCorriente(saldo=1000, tasa_anual=2.5, sobregiro=200)
cuenta3 = CuentaAhorros(saldo=0, tasa_anual=4.0, activa=False)

# Operaciones con las cuentas
cuenta1.consignar(200)
cuenta1.retirar(100)
cuenta1.extracto_mensual()
cuenta1.imprimir()

cuenta2.retirar(1200)
cuenta2.extracto_mensual()
cuenta2.imprimir()

cuenta3.consignar(50)
cuenta3.retirar(20)
cuenta3.extracto_mensual()
cuenta3.imprimir()

```

## CAPTURA

Instanciando 3 objetos.

```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> py .\ejercicio1.py
Consignación exitosa. Nuevo saldo: 700
Retiro exitoso. Nuevo saldo: 600
Extracto mensual aplicado. Saldo final: 600.0

Saldo: 600.0
Número de consignaciones: 1
Número de retiros: 1
Tasa anual: 3.5%
Comisión mensual: 0.0

Cuenta Activa: Sí

Extracto mensual aplicado. Saldo final: -400.0

Saldo: -400.0
Número de consignaciones: 0
Número de retiros: 0
Tasa anual: 2.5%
Comisión mensual: 0.0

Saldo de sobregiro: 0

Consignación exitosa. Nuevo saldo: 50
Retiro exitoso. Nuevo saldo: 30
Extracto mensual aplicado. Saldo final: 30.0

Saldo: 30.0
Número de consignaciones: 1
Número de retiros: 1
Tasa anual: 4.0%
Comisión mensual: 0.0

Cuenta Activa: No

PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1>
```

## Aplicando principio de herencia

```
# Instanciación de objetos
cuenta1 = CuentaAhorros(saldo=500, tasa_anual=3.5, activa=True)
cuenta2 = CuentaCorriente(saldo=1000, tasa_anual=2.5, sobregiro=200)
cuenta3 = CuentaAhorros(saldo=0, tasa_anual=4.0, activa=False)

# Operaciones con las cuentas
cuenta1.consignar(200)
cuenta1.retirar(100)
cuenta1.extracto_mensual()
cuenta1.imprimir()

cuenta2.retirar(1200)
cuenta2.extracto_mensual()
cuenta2.imprimir()

cuenta3.consignar(50)
cuenta3.retirar(20)
cuenta3.extracto_mensual()
cuenta3.imprimir()
```

20. Sean Colegio, Instituto Técnico y Universidad, entre instituciones educativas públicas y privadas.

|                                |
|--------------------------------|
| Colegio                        |
| Nombre: String                 |
| Tipo: String (Privado/Público) |
| NroProfesores: int             |
| nroEstudiantes: int            |
| turno: String                  |
| MostrarColegio()               |

|                                |
|--------------------------------|
| Instituto                      |
| Nombre: String                 |
| Tipo: String (Privado/Público) |
| NroProfesores: int             |
| nroEstudiantes: int            |
| nroEspecialidades: int         |
| MostrarInstituto()             |

|                                |
|--------------------------------|
| Universidad                    |
| Nombre: String                 |
| Tipo: String (Privado/Público) |
| NroProfesores: int             |
| nroEstudiantes: int            |
| nroCarreras: int               |
| MostrarUniversidad()           |

- Instanciar 3 objetos de cada una de las clases.
- Mostrar un colegio, un instituto y una universidad.
- Aplicar el principio de Herencia.
- Aplicar encapsulamiento en los atributos y métodos.

## CÓDIGO

### Clase Padre Institución con atributos.

```
class InstitucionEducativa:
    def __init__(self, nombre, tipo, nro_profesores, nro_estudiantes):
        self.__nombre = nombre
        self.__tipo = tipo
        self.__nro_profesores = nro_profesores
        self.__nro_estudiantes = nro_estudiantes

    def get_nombre(self):
        return self.__nombre

    def get_tipo(self):
        return self.__tipo

    def get_nro_profesores(self):
        return self.__nro_profesores

    def get_nro_estudiantes(self):
        return self.__nro_estudiantes

    def mostrar_info(self):
        print(f"Nombre: {self.__nombre}")
        print(f"Tipo: {self.__tipo}")
        print(f"Número de Profesores: {self.__nro_profesores}")
        print(f"Número de Estudiantes: {self.__nro_estudiantes}")
```

## Clase Hija universidad

```
from ejercicio20_institucion import InstitucionEducativa

class Universidad(InstitucionEducativa):
    def __init__(self, nombre, tipo, nro_profesores, nro_estudiantes, nro_carreras):
        super().__init__(nombre, tipo, nro_profesores, nro_estudiantes)
        self.__nro_carreras = nro_carreras

    def get_nro_carreras(self):
        return self.__nro_carreras

    def mostrar_info(self):
        super().mostrar_info()
        print(f"Número de Carreras: {self.__nro_carreras}")
        print("-----")
```

## Clase hija instituto

```
from ejercicio20_institucion import InstitucionEducativa

class Instituto(InstitucionEducativa):
    def __init__(self, nombre, tipo, nro_profesores, nro_estudiantes, nro_especialidades):
        super().__init__(nombre, tipo, nro_profesores, nro_estudiantes)
        self.__nro_especialidades = nro_especialidades

    def get_nro_especialidades(self):
        return self.__nro_especialidades

    def mostrar_info(self):
        super().mostrar_info()
        print(f"Número de Especialidades: {self.__nro_especialidades}")
        print("-----")
```

## Clase hija colegio

```
from ejercicio20_institucion import InstitucionEducativa

class Colegio(InstitucionEducativa):
    def __init__(self, nombre, tipo, nro_profesores, nro_estudiantes, turno):
        super().__init__(nombre, tipo, nro_profesores, nro_estudiantes)
        self.__turno = turno

    def get_turno(self):
        return self.__turno

    def mostrar_info(self):
        super().mostrar_info()
        print(f"Turno: {self.__turno}")
        print("-----")
```



## CLASE MAIN INSTANCIANDO UN OBJETO POR CADA CLASE HIJA Y APLICANDO HERENCIA EN LOS ATRIBUTOS Y METODOS

```
from ejercicio20_colegio import Colegio
from ejercicio20_instituto import Instituto
from ejercicio20_universidad import Universidad

# Instanciar 3 objetos de cada clase
colegio1 = Colegio("Colegio ABC", "Privado", 30, 500, "Mañana")
colegio2 = Colegio("Colegio XYZ", "Público", 40, 600, "Tarde")
colegio3 = Colegio("Colegio Delta", "Privado", 25, 450, "Nocturno")

instituto1 = Instituto("Instituto Tecno", "Privado", 50, 800, 10)
instituto2 = Instituto("Instituto Pro", "Público", 40, 700, 8)
instituto3 = Instituto("Instituto Sigma", "Privado", 60, 900, 12)

universidad1 = Universidad("Universidad Nacional", "Público", 200, 5000, 30)
universidad2 = Universidad("Universidad Privada", "Privado", 150, 4000, 25)
universidad3 = Universidad("Universidad Estatal", "Público", 180, 4500, 28)

# Mostrar un colegio, un instituto y una universidad
print("=== Información de un Colegio ===")
colegio1.mostrar_info()

print("=== Información de un Instituto ===")
instituto1.mostrar_info()

print("=== Información de una Universidad ===")
universidad1.mostrar_info()
```

## CAPTURA

Mostrando 3 objetos por cada clase hija:

```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> py .\e
=== Información de un Colegio ===
Nombre: Colegio ABC
Tipo: Privado
Número de Profesores: 30
Número de Estudiantes: 500
Turno: Mañana
-----
=== Información de un Instituto ===
Nombre: Instituto Tecno
Tipo: Privado
Número de Profesores: 50
Número de Estudiantes: 800
Número de Especialidades: 10
-----
=== Información de una Universidad ===
Nombre: Universidad Nacional
Tipo: Público
Número de Profesores: 200
Número de Estudiantes: 5000
Número de Carreras: 30
-----
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\PRACTICA1> |
```