**UNIVERSIDAD MAYOR DE SAN ANDRÉS**

**FACULTAD DE TECNOLOGÍA**

**ELECTRÓNICA Y TELECOMUNICACIONES**

***Logotipo

El contenido generado por IA puede ser incorrecto.***

***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 dia : "))

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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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("Ingrese 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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("Ingrese numero: "))

    if n > 0:

        print(f"El cudrado es: {n\*n}")

    else:

        break

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

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

Ejemplo: 962 9 + 6 + 2 = 17

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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Imagen de la pantalla de un celular de un mensaje en letras blancas

El contenido generado por IA puede ser incorrecto.

1. 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'.

Imagen que contiene lado, grande, hombre, teclado

El contenido generado por IA puede ser incorrecto.

**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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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.

Imagen que contiene Aplicación

El contenido generado por IA puede ser incorrecto.

**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**

Calendario

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

1. Realizar un programa aplicando la POO de la clase Articulo científico:

Tabla

El contenido generado por IA puede ser incorrecto.

1. Instanciar N objetos y almacenarlos en una lista.

**CÓDIGO**

class ArticuloCientifico:

    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 = ArticuloCientifico(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**

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

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

Imagen que contiene Tabla

El contenido generado por IA puede ser incorrecto.

1. 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**

Texto

El contenido generado por IA puede ser incorrecto.

**Impresión de datos y búsqueda de un contacto.**

Texto

El contenido generado por IA puede ser incorrecto.

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

Diagrama

El contenido generado por IA puede ser incorrecto.

1. Instanciar 3 objetos
2. 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.**

Texto

El contenido generado por IA puede ser incorrecto.

**Aplicando principio de herencia**

Texto

El contenido generado por IA puede ser incorrecto.

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

Texto, Carta

El contenido generado por IA puede ser incorrecto.

1. Instanciar 3 objetos de cada una de las clases.
2. Mostrar un colegio, un instituto y una universidad.
3. Aplicar el principio de Herencia.
4. 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 INSATANCIANDO 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:

Texto

El contenido generado por IA puede ser incorrecto.