**UNIVERSIDAD MAYOR DE SAN ANDRÉS**

**FACULTAD DE TECNOLOGÍA**

**ELECTRÓNICA Y TELECOMUNICACIONES**

***Logotipo

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

***PRACTICA AUXILIATURA 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

**PARTE I - ESTRUCTURAS CONDICIONALES**

1. El promedio de prácticas de un curso se calcula en base a cuatro prácticas calificadas, de las cuales se elimina la nota menor y se promedian las tres notas más altas. Elabore un algoritmo que imprima: la nota eliminada y el promedio de prácticas de un estudiante.

Ejemplo: Entrada → 9 8 7 10 Salida → Nota eliminada: 7 Promedio: 9

**CÓDIGO**

def ordenar(vector):

    n = len(vector)

    for i in range(n):

        for j in range(0, n - i - 1):

            if vector[j] > vector[j + 1]:

                vector[j], vector[j + 1] = vector[j + 1], vector[j]

    return vector

#Main

vector = list(map(int, input().split()))

vector\_ordenado = ordenar(vector)

promedio = (vector\_ordenado[1]+vector\_ordenado[2]+vector\_ordenado[3])//3

print(f"Nota eliminada: {vector\_ordenado[0]} Promedio: {promedio}")

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Verificar si la fecha proporcionada es correcta. La entrada consiste de tres números enteros a, b, c. Representando el día, mes y año. Ejemplo:

Entrada: Salida:

29 02 2003 Fecha Incorrecta

30 07 2003 Fecha Correcta

05 13 1991 Fecha Incorrecta

**CÓDIGO**

def fecha\_valida(dia, mes, año):

    dias\_mes = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

    if mes < 1 or mes > 12:

        return "Fecha Incorrecta"

    if (año % 4 == 0 and año % 100 != 0) or (año % 400 == 0):

        dias\_mes[1] = 29

    if dia < 1 or dia > dias\_mes[mes - 1]:

        return "Fecha Incorrecta"

    return "Fecha Correcta"

# MAIN

a, b, c = map(int, input().split())

print(fecha\_valida(a, b, c))

a, b, c = map(int, input().split())

print(fecha\_valida(a, b, c))

a, b, c = map(int, input().split())

print(fecha\_valida(a, b, c))

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Escriba un programa que tome como entrada un número M donde 1 ≤ M ≤ 12 y devuelva el nombre del mes correspondiente en español.

Ejemplo:

Entrada: Salida:

1 enero

11 noviembre

**CÓDIGO**

def dev\_mes(numero):

    if m >= 1 or m <= 12:

        if m == 1:

            return "Enero"

        elif m == 2:

            return "Febrero"

        elif m == 3:

            return "Marzo"

        elif m == 4:

            return "Abril"

        elif m == 5:

            return "Mayo"

        elif m == 6:

            return "Junio"

        elif m == 7:

            return "Julio"

        elif m == 8:

            return "Agosto"

        elif m == 9:

            return "Septiembre"

        elif m == 10:

            return "Octubre"

        elif m == 11:

            return "Noviembre"

        elif m == 12:

            return "Diciembre"

    return "No valido"

#Main

m = int(input())

print(dev\_mes(m))

m = int(input())

print(dev\_mes(m))

**CAPTURA**

Imagen que contiene foto, medidor, oscuro, hombre

El contenido generado por IA puede ser incorrecto.

1. FRealizar un programa 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. Recibir N número enteros, entre los cuales hay positivos, negativos y ceros. Se debe calcular el promedio de los que fueran simultáneamente positivo y múltiplos de 5. Si no hubiera ningún número con estés condiciones, mostrar un mensaje “inexistente”.

**CÓDIGO**

n = int(input("Ingrese la cantidad de numeros: "))

suma, contador = 0, 0

for \_ in range(n):

    valor = int(input())

    if (valor > 0) and (valor % 5 == 0):

        suma += valor

        contador += 1

if contador != 0:

    print(f"El promedio es {suma/contador}")

else:

    print("inexistente")

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

**PARTE II - ESTRUCTURAS REPITITIVAS**

1. DElabore un algoritmo que dado un número N, halle la suma de los números pares e impares, muestre por pantalla el resultado de ambas sumas.

Ejemplo:

Entrada: |Salida:

10 30

25

**CÓDIGO**

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

pares, impares = 0, 0

for i in range(1, n+1):

    if i % 2 == 0:

        pares += i

    else:

        impares += i

print("pares:", pares)

print("impares:", impares)

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Escribir un programa para mostrar por pantalla la siguiente pirámide de dígitos para un valor de n entre 1 y 9 (validarlo).

Por ejemplo, para n=5:

12345

1234

123

12

1

**CÓDIGO**

while True:

    n = int(input("Ingrese N del 1 al 9:"))

    if n >=1 and n <= 9:

        break

for i in range(5, 0,-1):

    for j in range(1,i+1):

        print(j, end = "")

    print()

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dado un número natural, generar:

Ejemplo, para n=5:

12345

23451

34512

45123

51234

**CÓDIGO**

cadena = ""

n = int(input("ingrese un numero del 1 al 9: "))

for \_ in range(1, n+1):

    cadena += str(\_)

for i in range(n):

        print(cadena)

        cadena = cadena[1:] + cadena[0]

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dado un número natural, generar:

Ejemplo, para n=5:

1

121

12321

1234321

123454321

5

5

5

**CÓDIGO**

n = int(input("ingrese n: "))

for i in range(1, n + 1):

    # Crear I>zquierda (de 1 hasta i)

    ascendente = ""

    for j in range(1, i + 1):

        ascendente += str(j)

    # Crear derecha(de i-1 hasta 1)

    descendente = ""

    for j in range(i - 1, 0, -1):

        descendente += str(j)

    # Espácios Izq

    espacios = " " \* (n - i)

    # Imprimir

    print(espacios + ascendente + descendente)

print(" " \* (n - 1) + str(n))

print(" " \* (n - 1) + str(n))

print(" " \* (n - 1) + str(n))

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Realizar un programa que realice el siguiente datagrama con un N introducido por teclado, por ejemplo: Para un N=5

Un reloj digital

El contenido generado por IA puede ser incorrecto.

**CÓDIGO**

n = int(input("ingresa n: "))

for i in range(n):

    for j in range(n):

        if (i+j) == (n-1):

            print("1",end = " ")

        elif (i == 1 and j == 0) or (i == (n-2) and j == (n-1)):

             print("0",end = " ")

        elif (i+j) == (n-1):

            print("1", end = " ")

        elif i == 0:

            print("0",end = " ")

        elif n-1 == i:

             print("0",end = " ")

        else:

            print(" ", end = " ")

    print()

**CAPTURA**

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**PARTE III - LISTAS Y DIRECCIONARIOS**

1. Dada una cadena se pide eliminar las vocales y duplicar las consonantes.

Ejemplo:

Entrada: Salida:

caminante no hay camino ccmmnnnntt nn hhyy ccmmnn

hola mundo hhll mmnndd

**CÓDIGO**

x = input("Ingrese la cadena: ")

cadena = ""

for i in range(len(x)):

    if x[i] == " ":

        cadena += " "

    elif x[i].lower() not in "aeiouáéíóúàèìòùäëïöü":

        cadena = cadena + x[i] + x[i]

print(cadena)

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dada una lista de palabras, contar cuántas vocales hay en total.

**CÓDIGO**

palabras = []

n = int(input("cantidad de palabras: "))

c =0

for \_ in range(n):

    palabras.append(input())

for i in palabras:

    for j in range(len(i)):

        if i[j].lower() in "aeiouáéíóúàèìòùäëïöü":

            c += 1

print(f"total vocales: {c}")

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dados dos listas: nombres y edades, combinar ambas y encontrar el tercer nombre con la menor edad.

**CÓDIGO**

nombres = ["Ana", "Luis", "Carlos", "Maria", "Pedro"]

edades = [23, 30, 18, 25, 22]

print("nombres", nombres)

print("edades: ", edades)

for i in range(len(edades)):

    for j in range(0, len(edades) - i - 1):

        if edades[j] > edades[j + 1]:

            edades[j], edades[j + 1] = edades[j + 1], edades[j]

            nombres[j], nombres[j + 1] = nombres[j + 1], nombres[j]

#main

print()

print("ordenando")

print("nombres", nombres)

print("edades: ", edades)

tercer\_nombre = nombres[2]

# Mostrar el resultado

print(f"El tercer nombre con la menor edad es: {tercer\_nombre}")

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Convertir 'a' → 1, 'e' → 2, 'i' → 3, 'o' → 4, 'u' → 5 en cada palabra de una lista.

**CÓDIGO**

palabras = []

n = int(input("Numero de palabras: "))

for \_ in range(n):

    palabras.append(input())

print(palabras)

c = 0

for i in palabras:

    palabra = ""

    for j in range(len(i)):

        if i[j] == " ":

            palabra += " "

        elif i[j] == "a":

            palabra += "1"

        elif i[j] == "e":

            palabra += "2"

        elif i[j] == "i":

            palabra += "3"

        elif i[j] == "o":

            palabra += "4"

        elif i[j] == "u":

            palabra += "5"

        else:

            palabra += i[j]

    palabras[c] = palabra

    c += 1

print(palabras)

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dadas dos listas de números, sumarlas, mostrar la lista resultante y de esta misma obtener los múltiplos de 3.

**CÓDIGO**

# Listas de números (pueden tener tamaños diferentes)

lista1 = [1, 2, 3, 4, 5, 6]

lista2 = [6, 7, 8, 9]

print("lista1:",lista1)

print("lista2:", lista2)

if len(lista2) > len(lista1):

    lista1, lista2 = lista2, lista1

for i in range(len(lista2)):

    lista1[i] += lista2[i]

print("suma de los vectores: ", lista1)

print("multiplos de 3: ")

for i in lista1:

    if i % 3 == 0:

        print(i)

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dado un diccionario, eliminar las claves que comiencen con una vocal.

**CÓDIGO**

# Definir el diccionario

diccionario = {

    "apple": 10,

    "banana": 20,

    "orange": 15,

    "uva": 30,

    "mango": 25,

    "elefante": 40,

    "iguana": 50,

    "oso": 60

}

for v in diccionario.items():

    print(f"{v[0]}: {v[1]}")

vocales = ['a', 'e', 'i', 'o', 'u']

claves\_a\_eliminar = []

for clave in diccionario:

    if clave[0].lower() in vocales:

        claves\_a\_eliminar.append(clave)

for clave in claves\_a\_eliminar:

    del diccionario[clave]

print()

print("diccionario resultante: ")

print(diccionario)

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Realizar un programa que tome una cadena de texto y devuelva un diccionario donde las llaves sean las palabras y los valores la cantidad de veces que aparecen.

Entrada: "python es genial y python es poderoso"

Salida: {'python': 2, 'es': 2, 'genial': 1, 'y': 1, 'poderoso': 1}

**CÓDIGO**

diccionario = {}

palabras = input("ingrese la cadena: ").split(" ")

for i in palabras:

    if i in diccionario:

        diccionario[i] = diccionario[i] + 1

    else:

         diccionario[i] = 1

print(diccionario)

**CAPTURA**

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Si el valor es un número par, cambiarlo a 'Par'. Si es impar, cambiarlo a 'Impar'.

Entrada: lista\_numeros = [3, 8, 15, 22, 7, 10, 5]

Salida: ['Impar', 'Par', 'Impar', 'Par', 'Impar', 'Par', 'Impar']

**CÓDIGO**

numeros = []

n = int(input("ingresar numero n: "))

for \_ in range(n):

    numeros.append(int(input()))

print(numeros)

for i in range(len(numeros)):

    if numeros[i] % 2 == 0:

        numeros[i] = "Par"

    else:

        numeros[i] = "Impar"

print(numeros)

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dado un diccionario con palabras como valores, contar la frecuencia de cada vocal.

**CÓDIGO**

def contar\_vocales(diccionario):

    vocales = ["a", "e", "i", "o", "u", "A", "E", "I", "O", "U"]

    frecuencia = {

        "a": 0, "e": 0, "i": 0, "o": 0, "u": 0,

        "A": 0, "E": 0, "I": 0, "O": 0, "U": 0

    }

    for clave in diccionario:

        palabra = diccionario[clave]

        for letra in palabra:

            if letra in vocales:

                frecuencia[letra] += 1

    resultado = {}

    for vocal in frecuencia:

        if frecuencia[vocal] > 0:

            resultado[vocal] = frecuencia[vocal]

    return resultado

diccionario = {"uno": "hola", "dos": "mundo", "tres": "python"}

print("el diccionario:")

for i in diccionario.items():

    print(f"{i[0]} -> {i[1]}")

resultado = contar\_vocales(diccionario)

print()

print("Contador de vocales: ")

for i in resultado.items():

    print(f"{i[0]} -> {i[1]}")

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Dados dos diccionarios, combinarlos y ordenar sus claves.

**CÓDIGO**

def combinar\_diccionarios(dic1, dic2):

    combinado = {}

    for clave in dic1:

        combinado[clave] = dic1[clave]

    for clave in dic2:

        combinado[clave] = dic2[clave]

    claves = list(combinado.keys())

    n = len(claves)

    for i in range(n):

        for j in range(0, n - i - 1):

            if str(claves[j]) > str(claves[j + 1]):

                claves[j], claves[j + 1] = claves[j + 1], claves[j]

    diccionario\_ordenado = {}

    for clave in claves:

        diccionario\_ordenado[clave] = combinado[clave]

    return diccionario\_ordenado

#MAIN

dic1 = {3: "tres", "a": "letra A", 10: "diez", "z": "letra Z"}

dic2 = {"b": "letra B", 1: "uno", "c": "letra C", 5: "cinco", "@": "arroba"}

print("diccionario\_1")

for i in dic1.items():

    print(f"{i[0]}: {i[1]}")

print()

print("diccionario 2")

for i in dic1.items():

    print(f"{i[0]}: {i[1]}")

resultado = combinar\_diccionarios(dic1, dic2)

print()

print("diccionario union ordenado")

for i in resultado.items():

    print(f"{i[0]}: {i[1]}")

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

**PARTE IV - POO**

1. Usando programación orientada a objetos Construya una clase para efectuar las siguientes operaciones aritméticas de dos números ingresados diferentes de cero.

Texto

El contenido generado por IA puede ser incorrecto.

**CÓDIGO**

class Opercaciones:

    def \_\_init\_\_(self, p\_valor, s\_valor):

        self.primer\_valor = p\_valor

        self.segundo\_valor = s\_valor

    def potencia(self):

        print(f"podencia es igual a: {self.primer\_valor\*\*self.segundo\_valor}")

    def raiz\_cuadrada(self):

        print(f"raiz del 1er valor es igual a: {self.primer\_valor\*\*1/2}")

        print(f"raiz del 2do valor es igual a: {self.segundo\_valor\*\*1/2}")

    def  multiplicacion(self):

        print(f"multitiplicación es igual a: {self.primer\_valor\*self.segundo\_valor}")

    def division(self):

        if self.segundo\_valor == 0:

            print(f"denominador es 0 division indeterminado")

        else:

            print(f"division es igual a: {self.primer\_valor/self.segundo\_valor}")

#main

pv = float(input("primer valor: "))

ps = float(input("segundo valor: "))

op1 = Opercaciones(pv, ps)

op1.potencia()

op1.raiz\_cuadrada()

op1.multiplicacion()

op1.division()

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Crear la clase CuentaBancaria para que tenga un método transferir que permita transferir dinero entre dos cuentas.

Texto

El contenido generado por IA puede ser incorrecto.

**CÓDIGO**

class CuentaBancaria:

    def \_\_init\_\_(self, p\_cuenta,s\_pcuenta,s\_cuenta,s\_scuenta):

        self.primera\_cuenta = p\_cuenta

        self.saldo\_pcuenta = s\_pcuenta

        self.segunda\_cuenta = s\_cuenta

        self.saldo\_scuenta = s\_scuenta

    def transferencia\_entre\_cuentas(self):

        x = input("Ingrese la cuenta del cual quiere transferir: ")

        if x == self.primera\_cuenta:

            while True:

                m = float(input("Ingrese el monto a transferir: "))

                if m <= self.saldo\_pcuenta:

                    break

                else:

                    print("el monto es mayor al saldo disponible")

            self.saldo\_scuenta += m

            self.saldo\_pcuenta -= m

        elif x == self.segunda\_cuenta:

            while True:

                m = float(input("Ingrese el monto a transferir: "))

                if m <= self.saldo\_scuenta:

                    break

                else:

                    print("el monto es mayor al saldo disponible: ")

            self.saldo\_pcuenta += m

            self.saldo\_scuenta -= m

        else:

            print("Cuenta no reconocida")

    def mostrar(self):

       print(f"1° cuenta: {self.primera\_cuenta} saldo: {self.saldo\_pcuenta}")

        print(f"2° cuenta: {self.segunda\_cuenta} saldo: {self.saldo\_scuenta}")

#MAIN

cuenta = CuentaBancaria("2222",3500, "1111", 4000)

cuenta.mostrar()

print()

cuenta.transferencia\_entre\_cuentas()

print()

cuenta.mostrar()

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Crea una clase Coche con los atributos privados \_marca, \_modelo y \_velocidad. Implementa un método acelerar() que aumente la velocidad y frenar() que la disminuya. Luego, instancia un objeto de esta clase y realiza operaciones con él.

Ejemplo:

mi\_coche = Coche("Toyota", "Corolla")

mi\_coche.acelerar()

mi\_coche.frenar()

print(mi\_coche.obtener\_velocidad()) # 0

**CÓDIGO**

class Coche:

    def \_\_init\_\_(self, marca, modelo):

        self.\_marca = marca

        self.\_modelo = modelo

        self.\_velocidad = 0

    def acelerar(self):

        self.\_velocidad += 5

    def frenar(self):

        self.\_velocidad = max(0, self.\_velocidad - 5)

    def obtener\_velocidad(self):

        return self.\_velocidad

#MAIN

marca = input("Ingrese marca: ")

modelo = input("Ingrese modelo: ")

mi\_coche = Coche(marca, modelo)

mi\_coche.acelerar()

mi\_coche.frenar()

print("velocidad:", mi\_coche.obtener\_velocidad())

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Crea una clase base CuentaBancaria con atributos privados \_titular, \_saldo y métodos para depositar y retirar dinero. Luego, crea una subclase CuentaAhorro que tenga un atributo adicional \_tasa\_interes y un método aplicar\_interes() para aumentar el saldo según la tasa de interés.
2. Aplicar herencia y encapsulamiento. Donde lo heredado de la clase Padre debe ser: titular y saldo.

**CÓDIGO**

class CuentaBancaria:

    def \_\_init\_\_(self, titular, saldo\_inicial):

        self.\_titular = titular

        self.\_saldo = saldo\_inicial

    def depositar(self, cantidad):

        if cantidad > 0:

            self.\_saldo += cantidad

            print(f"Depósito realizado. Nuevo saldo: {self.\_saldo}")

        else:

            print("La cantidad a depositar debe ser mayor que 0.")

    def retirar(self, cantidad):

        if cantidad > 0 and cantidad <= self.\_saldo:

            self.\_saldo -= cantidad

            print(f"Retiro realizado. Nuevo saldo: {self.\_saldo}")

        else:

            print("Saldo insuficiente o cantidad no válida.")

    def obtener\_saldo(self):

        return self.\_saldo

class CuentaAhorro(CuentaBancaria):

    def \_\_init\_\_(self, titular, saldo\_inicial, tasa\_interes):

        super().\_\_init\_\_(titular, saldo\_inicial)

        self.\_tasa\_interes = tasa\_interes

    def aplicar\_interes(self):

        interes = self.\_saldo \* self.\_tasa\_interes / 100

        self.\_saldo += interes

        print(f"Interés aplicado. Nuevo saldo: {self.\_saldo}")

# MAIN

titular = input("ingresar titular: ")

inicial = int(input("ingresar saldo inicial: "))

interes = int(input("interes: "))

mi\_cuenta = CuentaAhorro(titular, inicial, interes)

print(f"Saldo es: {mi\_cuenta.obtener\_saldo()}")

print()

mi\_cuenta.depositar(int(input("ingrese monto a depositar: ")))

mi\_cuenta.retirar(int(input("ingrese monto a retirar: ")))

print()

print("aplicando interes")

mi\_cuenta.aplicar\_interes()

print(f"Saldo es: {mi\_cuenta.obtener\_saldo()}")

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

1. Crea una clase Animal con un método hacer\_sonido(). Luego, crea 4 subclases Perro Gato, Pato y Vaca que sobreescriban este método con sus respectivos sonidos ("Guau", "Miau", etc.). Además, implementa sobrecarga de métodos en Perro para que hacer\_sonido() pueda recibir un número de repeticiones del sonido.

**CÓDIGO**

class Animal:

    def hacer\_sonido(self):

        pass

class Perro(Animal):

    def hacer\_sonido(self, repeticiones = 1):

        print("Guau! " \* repeticiones)

class Gato(Animal):

    def hacer\_sonido(self):

        print("Miau!")

class Pato(Animal):

    def hacer\_sonido(self):

        print("Cuac!")

class Vaca(Animal):

    def hacer\_sonido(self):

        print("Mu!")

# MAIN

perro = Perro()

perro.hacer\_sonido(3)

print()

gato = Gato()

gato.hacer\_sonido()

print()

pato = Pato()

pato.hacer\_sonido()

print()

vaca = Vaca()

vaca.hacer\_sonido()

**CAPTURA**

Texto

El contenido generado por IA puede ser incorrecto.

**PARTE V - TEORÍA**

1. La Programación Orientada a Objetos (POO) es un paradigma de programación que organiza el código en **CLASES** y **OBJETOS** para representar entidades del mundo real y sus interacciones.
2. Una clase en Python se define utilizando la palabra clave **class**.
3. ¿Qué es instanciar? Incluya un ejemplo de instanciación

class Persona:

    def \_\_init\_\_(self, nombre, edad):

        self.nombre = nombre

        self.edad = edad

persona1 = Persona("Juan", 30)

1. El método \_\_init\_\_ en Python se utiliza para **INICIALIZAR** cuando se crea una nueva instancia de una clase.
2. Los atributos en una clase son **VARIABLES** que representan las características o propiedades de un objeto.

**Preguntas Falso o Verdadero**

1. Un objeto es una instancia de una clase, y puede tener atributos y métodos que le permitan realizar ciertas tareas.

**V** F

1. El polimorfismo permite que diferentes clases utilicen el mismo nombre de método, pero con implementaciones diferentes dependiendo de la clase en la que se encuentre.

**V** F

1. La herencia en POO permite que una clase derive de otra clase, heredando sus atributos y métodos sin necesidad de reescribir el código.

**V** F

**Preguntas de desarrollo**

1. **Explica qué es el encapsulamiento y cómo se logra en Python. Da un ejemplo donde se utilicen métodos getter y setter para controlar el acceso a un atributo.**

**R:** El **encapsulamiento** es un principio de la programación orientada a objetos que se refiere a ocultar los detalles internos de una clase y exponer solo lo necesario. En Python, esto se logra mediante el uso de métodos getter y setter para acceder a los atributos privados de una clase. Ejemplo:

class Persona:

    def \_\_init\_\_(self, nombre):

        self.\_\_nombre = nombre  # Atributo privado

    def get\_nombre(self):

        return self.\_\_nombre  # Método getter

    def set\_nombre(self, nombre):

        if len(nombre) > 3:

            self.\_\_nombre = nombre  # Método setter

persona1 = Persona("Juan")

 # Acceso al atributo mediante getter

print(persona1.get\_nombre())

# Modificación del atributo mediante setter

persona1.set\_nombre("Carlos")

print(persona1.get\_nombre())

1. **Explique que es herencia**

**R:** es un mecanismo en la programación orientada a objetos que permite a una clase derivada (subclase) heredar atributos y métodos de una clase base (superclase). Esto permite la reutilización del código y la creación de jerarquías de clases. La subclase puede modificar o extender el comportamiento heredado.

1. **Explique que es polimorfismo en Python.**

**R:** El polimorfismo es un principio que permite que una misma interfaz sea utilizada para diferentes tipos de objetos. En Python, se puede lograr mediante el uso de métodos con el mismo nombre en diferentes clases, pero con implementaciones específicas para cada una.

1. **¿Qué es la abstracción en la Programación Orientada a Objetos?**

**R:** La abstracción la podemos definir como el proceso de identificar aquellas características (atributos) y acciones o comportamientos (métodos) propios de un elemento que deseemos representar.