

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



***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

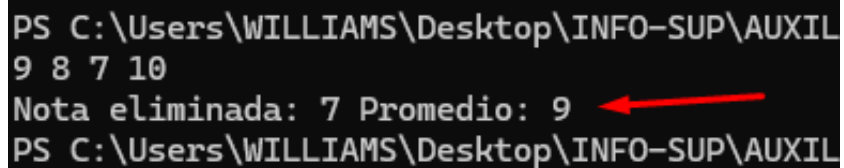
vector = list(map(int, input().split()))

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



```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXIL
9 8 7 10
Nota eliminada: 7 Promedio: 9
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXIL
```

2. 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(día, 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 día < 1 or día > dias_mes[mes - 1]:
        return "Fecha Incorrecta"

    return "Fecha Correcta"

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

print(fecha_valida(a, b, c))

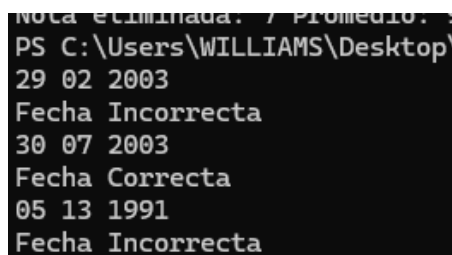
a, b, c = map(int, input().split())

print(fecha_valida(a, b, c))

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

print(fecha_valida(a, b, c))
```

CAPTURA



```
NOTA eliminada. / Promedio. 3
PS C:\Users\WILLIAMS\Desktop>
29 02 2003
Fecha Incorrecta
30 07 2003
Fecha Correcta
05 13 1991
Fecha Incorrecta
```

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

Ejemplo:

Entrada:

1

11

Salida:

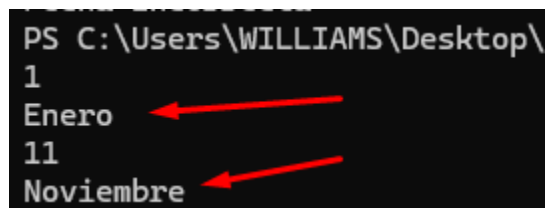
enero

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



```
PS C:\Users\WILLIAMS\Desktop\  
1  
Enero  
11  
Noviembre
```

4. Realizar 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 = []

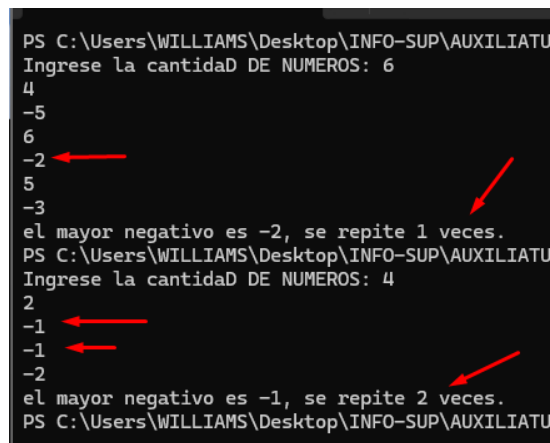
for _ in range(n):
    valor = int(input())
    if valor < 0:
        vector.append(valor)

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



```
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
```

5. 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 estas condiciones, mostrar un mensaje “inexistente”.

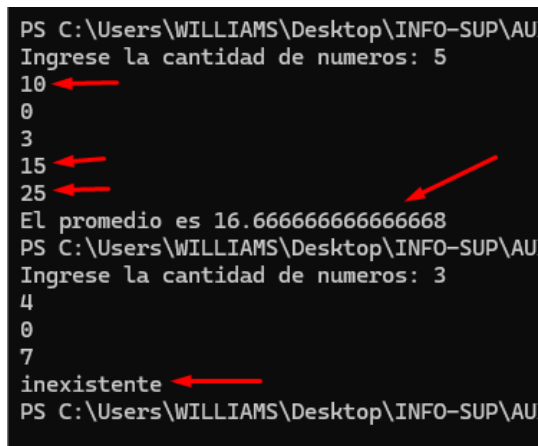
CÓDIGO

```
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



```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AU
Ingrese la cantidad de numeros: 5
10
0
3
15
25
El promedio es 16.666666666666668
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AU
Ingrese la cantidad de numeros: 3
4
0
7
inexistente
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AU
```

The screenshot shows a PowerShell terminal window. The first part shows the user entering '5' for the number of inputs. Then, five numbers are entered: 10, 0, 3, 15, and 25. The program calculates the average of the positive multiples of 5 (15 and 25), which is 16.666666666666668. The second part shows the user entering '3' for the number of inputs. Then, three numbers are entered: 4, 0, and 7. Since none of these are positive multiples of 5, the program outputs 'inexistente'. Red arrows point to the input '5', the input '10', the input '15', the input '25', the output 'El promedio es 16.666666666666668', the input '3', the input '4', and the output 'inexistente'.

PARTE II - ESTRUCTURAS REPETITIVAS

1. Elabore 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

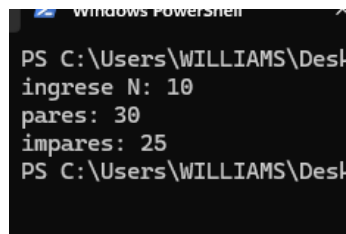
```
n = int(input("ingrese N: "))

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



```
PS C:\Users\WILLIAMS\Desktop> python3 programa1.py
ingrese N: 10
pares: 30
impares: 25
PS C:\Users\WILLIAMS\Desktop>
```

2. 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

```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\A
Ingrese N del 1 al 9:5
12345
1234
123
12
1
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\A
```

3. 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

```
PS C:\Users\WILLIAMS\Desktop\INFO-
ingrese un numero del 1 al 9: 5
12345
23451
34512
45123
51234
```

4. 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 Izquierda (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)

    # Espacios 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

```

Windows PowerShell
PS C:\Users\WILLIAMS\De
ingrese n: 5
1
121
12321
1234321
123454321
5
5
5
PS C:\Users\WILLIAMS\De

```

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

```

0 0 0 0 1
0      1
      1
    1      0
1 0 0 0 0

```

CÓDIGO

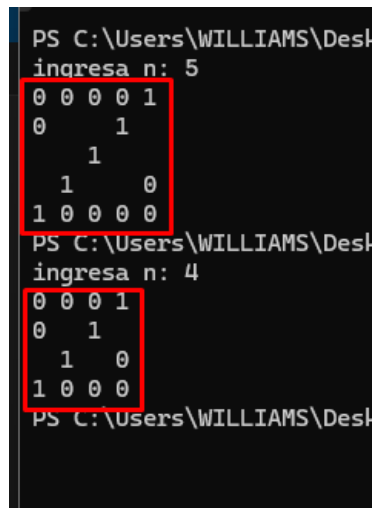
```
n = int(input("ingresa n: "))

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



```
PS C:\Users\WILLIAMS\Desktop>
ingresa n: 5
0 0 0 0 1
0      1
      1
    1    0
1 0 0 0 0
PS C:\Users\WILLIAMS\Desktop>
ingresa n: 4
0 0 0 1
0  1
  1  0
1 0 0 0
PS C:\Users\WILLIAMS\Desktop>
```

PARTE III - LISTAS Y DIRECCIONARIOS

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

Ejemplo:

Entrada:

caminante no hay camino

hola mundo

Salida:

ccmmnnnnntt nn hhyy ccmmnn

hhll mmnndd

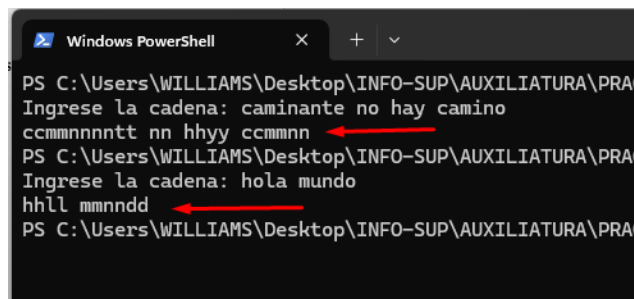
CÓDIGO

```
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



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

CÓDIGO

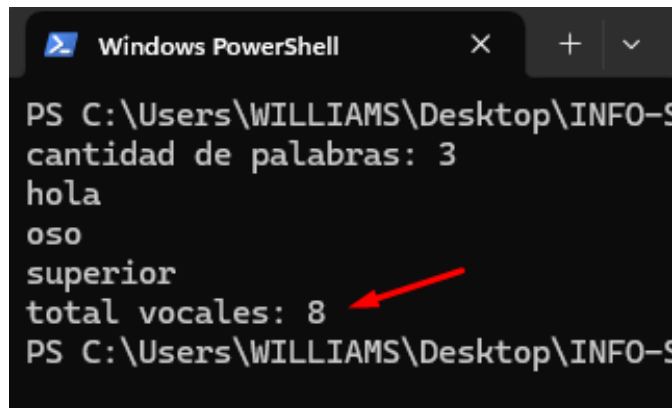
```
palabras = []
n = int(input("cantidad de palabras: "))
c = 0

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



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-S
cantidad de palabras: 3
hola
oso
superior
total vocales: 8
PS C:\Users\WILLIAMS\Desktop\INFO-S
```

3. 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

```
Instale la version mas reciente de PowerShell para obtener nuevas ca

PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> py .\36
nombres ['Ana', 'Luis', 'Carlos', 'Maria', 'Pedro']
edades:  [23, 30, 18, 25, 22]

ordenando
nombres ['Carlos', 'Pedro', 'Ana', 'Maria', 'Luis']
edades:  [18, 22, 23, 25, 30]
El tercer nombre con la menor edad es: Ana
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> |
```

4. 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

```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIAT...
Numero de palabras: 3
hola
oso
amigos nublire
['hola', 'oso', 'amigos nublire']
['h4l1', '4s4', '1m3g4s n5bl3r2']
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIAT...
```

5. 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

```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1
lista1: [1, 2, 3, 4, 5, 6]
lista2: [6, 7, 8, 9]
suma de los vectores: [7, 9, 11, 13, 5, 6]
multiplos de 3:
9
6
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1
```

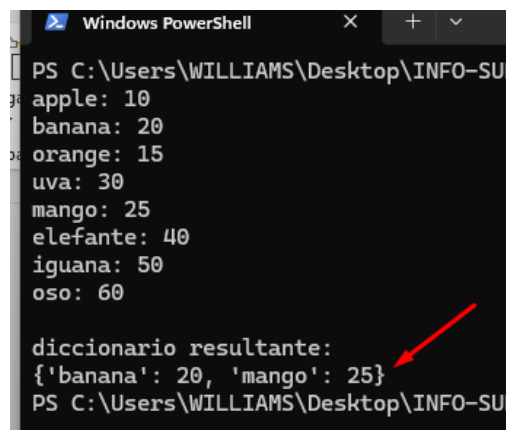
6. Dado un diccionario, eliminar las claves que comiencen con una vocal.

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



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SU
apple: 10
banana: 20
orange: 15
uva: 30
mango: 25
elefante: 40
iguana: 50
oso: 60

diccionario resultante:
{'banana': 20, 'mango': 25}
PS C:\Users\WILLIAMS\Desktop\INFO-SU
```

7. 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

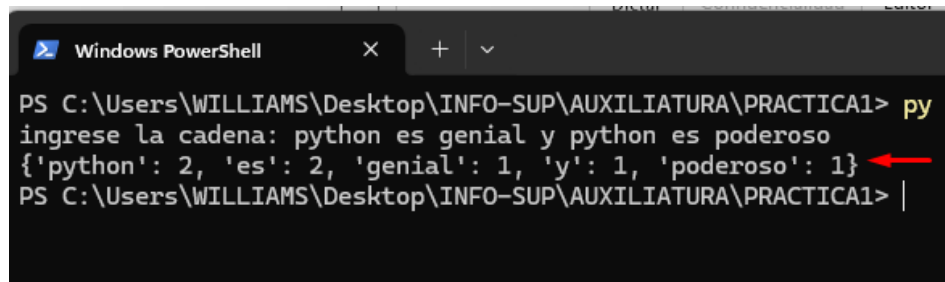
```
diccionario = {}

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



```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> py
ingrese la cadena: python es genial y python es poderoso
{'python': 2, 'es': 2, 'genial': 1, 'y': 1, 'poderoso': 1}
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> |
```

8. 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

```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> py .\3ejer
ingresar numero n: 7
3
8
15
22
7
10
5
[3, 8, 15, 22, 7, 10, 5]
['Impar', 'Par', 'Impar', 'Par', 'Impar', 'Par', 'Impar']
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> |
```

9. 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

```
PS C:\Users\WILLIAMS\Desktop\
el diccionario:
uno -> hola
dos -> mundo
tres -> python

Contador de vocales:
a -> 1
o -> 3
u -> 1
```

10. 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

```
PS C:\Users\WILLIAMS\Desktop\INFO-SUPV
diccionario_1 ←
3: tres
a: letra A
10: diez
z: letra Z

diccionario 2 ←
3: tres
a: letra A
10: diez
z: letra Z

diccionario union ordenado ←
1: uno
10: diez
3: tres
5: cinco
@: arroba
a: letra A
b: letra B
c: letra C
z: letra Z
PS C:\Users\WILLIAMS\Desktop\INFO-SUPV
```

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.

OPERACIONES
primer valor
segundo valor
Potencia ()
Raíz Cuadrada ()
Multiplicación ()
División ()

CÓDIGO

```
class Operaciones:

    def __init__(self, p_valor, s_valor):

        self.primer_valor = p_valor
        self.segundo_valor = s_valor

    def potencia(self):

        print(f"potencia 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"multiplicación es igual a: {self.primer_valor*self.segundo_valor}")

    def division(self):

        if self.segundo_valor == 0:

            print(f"denominador es 0 division indeterminado")

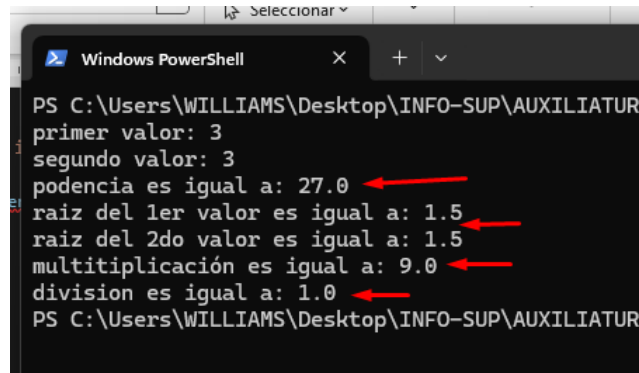
        else:

            print(f"division es igual a: {self.primer_valor/self.segundo_valor}")

#main
pv = float(input("primer valor: "))
ps = float(input("segundo valor: "))
op1 = Operaciones(pv, ps)

op1.potencia()
op1.raiz_cuadrada()
op1.multiplicacion()
op1.division()
```

CAPTURA



```
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATUR
primer valor: 3
segundo valor: 3
potencia es igual a: 27.0
raiz del 1er valor es igual a: 1.5
raiz del 2do valor es igual a: 1.5
multiplicación es igual a: 9.0
division es igual a: 1.0
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATUR
```

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

CUENTA BANCARIA	
Primera cuenta	Saldo cuenta
Segunda cuenta	Saldo cuenta
Transferencia entre cuentas	

CÓDIGO

```
class CuentaBancaria:

    def __init__(self, p_cuenta,s_pcuenta,s_cuenta,s_scuenta):
        self.primer_a_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.primer_a_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_scueta:
            break
        else:
            print("el monto es mayor al saldo disponible: ")
            self.saldo_pcueta += m
            self.saldo_scueta -= m
    else:
        print("Cuenta no reconocida")
def mostrar(self):
    print(f"1° cuenta: {self.primer_cuenta} saldo: {self.saldo_pcueta}")
    print(f"2° cuenta: {self.segunda_cuenta} saldo: {self.saldo_scueta}")

#MAIN
cuenta = CuentaBancaria("2222",3500, "1111", 4000)
cuenta.mostrar()
print()
cuenta.transferencia_entre_cuentas()
print()
cuenta.mostrar()

```

CAPTURA

```

Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> py .\4ejer
1° cuenta: 2222 saldo: 3500
2° cuenta: 1111 saldo: 4000

Ingrese la cuenta del cual quiere transferir: 1111
Ingrese el monto a transferir: 300

1° cuenta: 2222 saldo: 3800.0
2° cuenta: 1111 saldo: 3700.0
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTICA1> |

```

3. 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

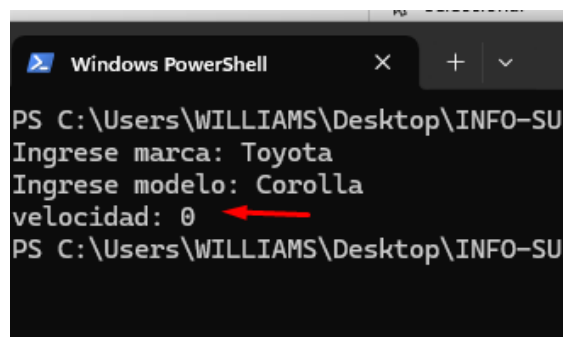
#MAIN

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



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SU
Ingrese marca: Toyota
Ingrese modelo: Corolla
velocidad: 0
PS C:\Users\WILLIAMS\Desktop\INFO-SU
```

4. 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.
 - a. 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.")

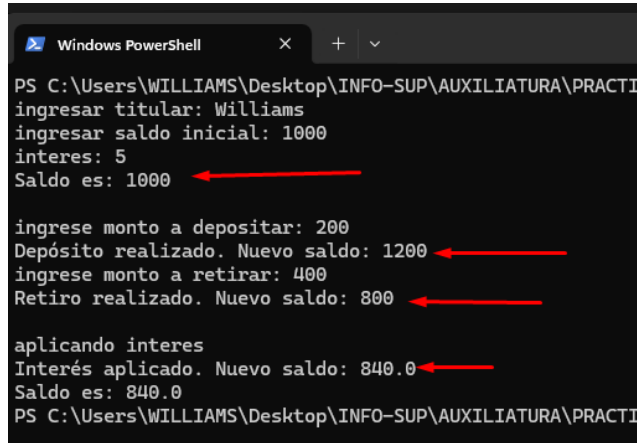
    def obtener_saldo(self):
        return self._saldo

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



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTI
ingresar titular: Williams
ingresar saldo inicial: 1000
interes: 5
Saldo es: 1000
ingrese monto a depositar: 200
Depósito realizado. Nuevo saldo: 1200
ingrese monto a retirar: 400
Retiro realizado. Nuevo saldo: 800
aplicando interes
Interés aplicado. Nuevo saldo: 840.0
Saldo es: 840.0
PS C:\Users\WILLIAMS\Desktop\INFO-SUP\AUXILIATURA\PRACTI
```

5. Crea una clase `Animal` con un método `hacer_sonido()`. Luego, crea 4 subclases `Perro`, `Gato`, `Pato` y `Vaca` que sobrescriban 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!")

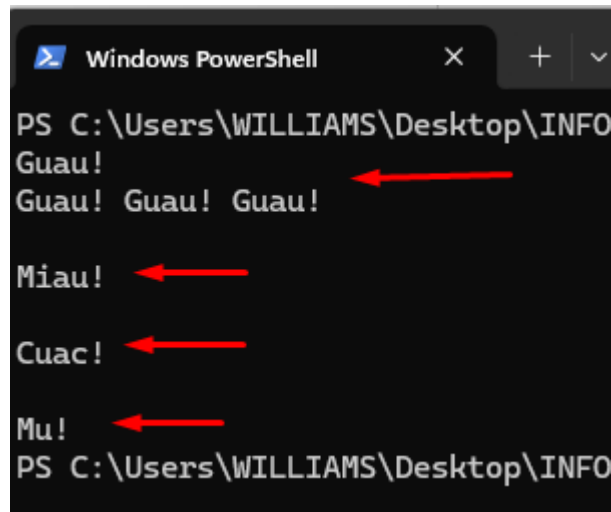
class Pato(Animal):
    def hacer_sonido(self):
        print("Cuac!")

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



```
Windows PowerShell
PS C:\Users\WILLIAMS\Desktop\INFO
Guau!
Guau! Guau! Guau!
Miau!
Cuac!
Mu!
PS C:\Users\WILLIAMS\Desktop\INFO
```

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? Incluye un ejemplo de instanciación

```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

persona1 = Persona("Juan", 30)
```

4. El método `__init__` en Python se utiliza para **INICIALIZAR** cuando se crea una nueva instancia de una clase.
5. Los atributos en una clase son **VARIABLES** que representan las características o propiedades de un objeto.

Preguntas Falso o Verdadero

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

V

F

7. 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

8. 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

9. 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())
```

10. 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.

11. 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.

12. ¿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.