TP 2 - Programación II

> **Note**
>
> IES 6023
> Docente: Lic. Arroyo, Carlos
> Cátedra: Programación II
> Alumno: Sandoval, José David Fernando

```python
"""
Ejercicio Nro 1
Realizar un programa que muestre la frase “Curso Python” 7 veces.
"""

def ingresar_frase() -> str:
    return "Curso Python"

def main():
    print(ingresar_frase())

if __name__ == "__main__":
    veces = 0
    while(veces < 7):
        main()
        veces += 1
```

```python
"""
Ejercicio Nro 2
Realizar un programa que permita ingresar números enteros,
mostrar la suma de números pares, y la cantidad de números impares.
El ingreso finaliza cuando el número es múltiplo de 5 (se incluye en el conteo).
"""

def ingresar_numero_entero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero entero: "))
        except ValueError:
            print("Debe ingresar valores correctos!")

def determinar_paridad(numero: int) -> bool:
    return numero % 2 == 0


def main():
    suma_pares = 0
    cantidad_impares = 0


    while True:
        numero = ingresar_numero_entero()
        if determinar_paridad(numero):
```

```
            suma_pares += 1
        else:
            cantidad_impares += 1

        if numero % 5 == 0:
            break

    print(f"La suma de numeros pares es: {suma_pares}")
    print(f"La cantidad de numeros impares es: {cantidad_impares}")

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 3
Realizar un programa que calcule el promedio de los estudiantes,
para ello se debe ingresar el apellido, nombre del estudiante y su nota.
Considere que el ingreso finaliza a pedido del operador.
Luego se debe mostrar el promedio
seguido de la cantidad de personas que ingresaron su apellido todo en mayúsculas.
"""

def ingresar_datos_estudiantes() -> tuple[str, str, float]:
    while True:
        try:
            apellido = input("Ingrese apellido: ")
            nombre = input("Ingrese nombre: ")
            nota = float(input("Ingrese nota: "))

            return apellido, nombre, nota

        except ValueError:
            print("Debe ingresar valores validos!")

def es_apellido_mayusculas(apellido: str) -> bool:
    return apellido.isupper()

def calcular_promedio(suma: float, cantidad: int) -> float:
    if cantidad == 0:
        return 0
    return suma / cantidad

def mostrar_resultado(promedio: float, cantidad_apellidos_mayusculas : int) -> None:
    print(f"\nEl promedio de los alumnos es: {promedio}")
    print(f"La cantidad de apellidos en mayusculas es: {cantidad_apellidos_mayusculas}")


def main():
    suma_notas = 0
    cantidad_estudiantes = 0
    cantidad_apellidos_mayusculas = 0

    while True:
        apellido, nombre, nota = ingresar_datos_estudiantes()

        suma_notas += nota
        cantidad_estudiantes += 1
```

```
        if es_apellido_mayusculas(apellido):
            cantidad_apellidos_mayusculas += 1

        respuesta = input("Desea finalizar la carga?(S/N): ")
        if respuesta.upper() == 'S':
            break

    promedio = calcular_promedio(suma_notas, cantidad_estudiantes)
    mostrar_resultado(promedio, cantidad_apellidos_mayusculas)

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 4
Realizar un programa que permita ingresar 16 números,
contar la cantidad de números negativos y cero,
y concatenar los números positivos en una variable a mostrar.
Salida esperada
Negativos: 5
Ceros: 3
Positivos: 32615144
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese valor entero: "))
        except:
            print("Debe ingresar un valor valido!")

def es_negativo(numero: int) -> bool:
    return numero < 0

def es_cero(numero: int) -> bool:
    return numero == 0

def es_positivo(numero: int) -> bool:
    return numero > 0


def main():
    negativos = 0
    ceros = 0
    concatenados = ""

# Repetimos el ingreso de números 16 veces,
# no necesitamos usar la variable del for

    for _ in range(16): # Va de de 0 hasta 16-1 (en este caso)
        num = ingresar_numero()
        if es_negativo(num):
            negativos += 1
        elif es_cero(num):
            ceros += 1
        elif es_positivo(num):
```

```
            concatenados += str(num)

    print(f"Negativos: {negativos}")
    print(f"Ceros: {ceros}")
    print(f"Positivos: {concatenados}")

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 5
Realizar un programa que permita ingresar letras
y por cada una contar las ocurrencias de las letras
del abecedario en mayúsculas,
también contar las letras minúsculas y dígitos.
El ciclo finaliza al ingresar un espacio.
Al finalizar muestre el total de letras ingresadas,
en mayúsculas, en minúsculas, la cantidad de vocales
que se ingresaron en mayúscula y minúscula,
la cantidad de letras 'J' ingresadas
y la cantidad de dígitos.
"""

def crear_contadores() -> dict:
    """
    Inicializa y devuelve un diccionario con todos los contadores necesarios.
    """

    # Esta línea crea un diccionario con las letras de la A a la Z como claves
    # y 0 como valor inicial para cada una de ellas.
    # chr(letra) convierte el número (código ASCII) en letra.
    # range(ord('A'), ord('Z') + 1) genera los códigos ASCII desde 'A' hasta 'Z'.
    ocurrencias_mayusculas = {chr(letra): 0 for letra in range(ord('A'), ord('Z') + 1)}

    return {
        "mayusculas": 0,
        "minusculas": 0,
        "digitos": 0,
        "vocales_mayus": 0,
        "vocales_minus": 0,
        "cantidad_j": 0,
        "ocurrencias_mayusculas": ocurrencias_mayusculas
    }

def ingresar_letra() -> str:
    while True:
        letra = input("Ingrese una letra (ESPACIO para finalizar): ")
        if len(letra) == 1:
            return letra
        print("Debe ingresar un solo caracter.")


def es_vocal(letra: str) -> bool:
    """
    Retorna True si la letra es una vocal.
    """
    return letra.lower() in "aeiou"
```

```python
def actualizar_contadores(letra: str, contadores: dict) -> None:
    """
    Recibe una letra y el diccionario de contadores.
    Actualiza los contadores según la letra ingresada.
    """

    if letra.isupper():
        contadores["mayusculas"] += 1
        contadores["ocurrencias_mayusculas"][letra] += 1
        if es_vocal(letra):
            contadores["vocales_mayus"] += 1

    elif letra.islower():
        contadores["minusculas"] += 1
        if es_vocal(letra):
            contadores["vocales_minus"] += 1

    elif letra.isdigit():
        contadores["digitos"] += 1

    if letra.upper() == 'J':
        contadores["cantidad_j"] += 1

def mostrar_resultados(contadores: dict) -> None:
    """
    Imprime todos los resultados almacenados en los contadores.
    """
    print(f"Total letras mayúsculas: {contadores['mayusculas']}")
    print(f"Total letras minúsculas: {contadores['minusculas']}")
    print(f"Total dígitos: {contadores['digitos']}")
    print(f"Vocales mayúsculas: {contadores['vocales_mayus']}")
    print(f"Vocales minúsculas: {contadores['vocales_minus']}")
    print(f"Cantidad de letras 'J': {contadores['cantidad_j']}")
    print("\nOcurrencias de letras mayúsculas:")

    # Recorremos el diccionario de ocurrencias. Cada par clave-valor (letra y su
cantidad) se separa automáticamente usando .items()

    for letra, cantidad in contadores["ocurrencias_mayusculas"].items():
        print(f"{letra}: {cantidad}")

def main() -> None:
    contadores = crear_contadores()

    while True:
        letra = ingresar_letra()
        if letra == " ":
            break
        actualizar_contadores(letra, contadores)

    mostrar_resultados(contadores)

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 6
Realizar un programa que permita ingresar números
y determinar si el número es par o no.
El ingreso de datos finaliza cuando se ingresa 999.
Determinar la cantidad de números pares
y la suma de impares ingresados.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def es_par(numero: int) -> bool:
    return numero % 2 == 0

def main() -> None:
    a = ingresar_numero()

    if a != 999:
        if es_par(a):
            print(f"{a} es par")
        else:
            print(f"{a} NO es par")

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 7
Realizar un programa que permita ingresar una frase
y la cantidad de veces que desea mostrar la frase.
Y mostrarla por pantalla esa cantidad de veces.
La frase debe mostrarse con cada palabra
empezando en mayúsculas para las iteraciones pares
(considere que la primera iteración es la 1).
"""

def ingresar_frase() -> str:
    while True:
        try:
            return input("Ingrese una frase: ")
        except ValueError:
            print("Ingrese un valor correcto!")

def ingresar_cantidad_de_repeticiones() -> int:
    while True:
        try:
            return int(input("Ingrese cantidad de repeticiones: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def main() -> None:
    frase = ingresar_frase()
```

```
    veces = ingresar_cantidad_de_repeticiones()

    for i in range(1, veces+1):
        # Si el número de repetición es par, aplicar .title()
        # para poner en mayúscula la primera letra de cada palabra.
        if i % 2 == 0:
            print(frase.title())
        else:
            # Si es impar, imprimir la frase tal como fue ingresada.
            print(frase)


if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 8
En matemáticas, particularmente en teoría de números o aritmética,
un número primo es un número natural mayor que 1
que tiene únicamente dos divisores distintos:
él mismo y el 1.
Diseñe un algoritmo que determine
si un valor ingresado por el usuario es primo o no.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un valor: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

"""
# Versión optimizada usando raíz cuadrada
import math

def es_primo(numero: int) -> bool:
    if numero < 2:
        return False

    for i in range(2, int(math.sqrt(numero)) + 1):
        if numero % i == 0:
            return False

    return True

"""


def es_primo(numero: int) -> bool:
    if numero < 2:
        return False # Por definicion, los num menores a 2 no son primos.
    for i in range(2, (int(numero/2)+ 1)):
        if numero % i == 0:
            return False # Si encontro un divisor, NO ES PRIMO
        return True # Si se da que no encontro un divisor, SI ES PRIMO
```

```python
def main() -> None:
    a = ingresar_numero()
    if es_primo(a):
        print(f"El numero {a} es primo")
    else:
        print(f"El numero {a} NO es primo!")

if __name__ == "__main__":
    main()
```

```python
"""
Ejercicio Nro 9
Realizar un programa que determine el mínimo de una serie
de números enteros ingresados por el usuario.
El ingreso finaliza cuando el valor introducido es 999.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Debe ingresar valores validos!")

# Determina el menor entre el minimo actual y el nuevo numero ingresado
def determinar_minimo(minimo: int, numero: int) -> int:

    if numero < minimo:
        return numero
    return minimo

def main() -> None:
    a = ingresar_numero()

    # Asumo que el primer numero es el minimo inicial
    minimo = a

    while a != 999:
        minimo = determinar_minimo(minimo, a)
        a = ingresar_numero()

    print(f"El minimo numero ingresado es: {minimo}")

if __name__ == "__main__":
    main()
```

```python
"""
Ejercicio Nro 10
Considerando que la potencia entre dos números enteros positivos, a y b,
la potencia puede expresarse como el producto sucesivo
de a tantas veces como lo indique b.
Pida ingresar estos dos valores,
y calcule la potencia con el algoritmo indicado.
"""

def ingresar_valores() -> tuple[int, int]:
    while True:
```

```
        try:
            a = int(input("Ingrese base: "))
            b = int(input("Ingrese exponente: "))
            return a, b
        except ValueError:
            print("Ingrese valores validos!")

def potencia(a: int, b: int) -> int:
    # 2^3 = 2 * 2 * 2
    pot = 1
    for _ in range(b):
        pot = pot * a
    return pot

def main() -> None:
    a, b = ingresar_valores()

    print(f"{a} elevado a {b} es igual a {potencia(a, b)}")

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 11
Realizar un programa que permita ingresar números,
hasta que el número ingresado no esté entre 1 y 100.
Determinar el promedio de los números ingresados entre 1 y 50.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Ingrese valores validos!")

def validar_numero_entre_1_y_100(numero: int) -> bool:
    return numero > 1 and numero < 100

def validar_numero_entre_1_y_50(numero: int) -> bool:
    return numero >= 1 and numero <= 50

def calcular_promedio(numero: int, suma: int, contador: int) -> tuple[int, int]:
    suma += numero
    contador += 1

    return suma, contador

def main() -> None:
    suma = 0
    contador = 0
    numero = ingresar_numero()

    while validar_numero_entre_1_y_100(numero):
        if validar_numero_entre_1_y_50(numero):
            suma, contador = calcular_promedio(numero, suma, contador)
```

```
        numero = ingresar_numero()

    if contador > 0:
        print(f"El promedio es: {suma / contador}")
    else:
        print("No se ingresaro numeros entre 1 y 50")

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 12
Realizar un programa donde se ingrese repetidamente números,
determinar si los números son pares.
El ingreso finaliza cuando el número es impar.
Use la sentencia break para salir del ciclo.
Al final mostrar el promedio de los números pares ingresados.
"""
from typing import Optional

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def es_par(numero: int) -> bool:
    return numero % 2 == 0


# Esta función devuelve un float si se puede calcular el promedio.
# Si no hay números pares (contador == 0), devuelve None.
# Usamos 'Optional[float]' para indicar que el resultado puede ser un float o None.
def determinar_promedio(contador: int, acumulador) -> Optional[float]:
    if contador == 0:
        return None
    return acumulador / contador


def main() -> None:
    contador_pares = 0
    acumulador_pares = 0


    while True:
        numero = ingresar_numero()
        if not es_par(numero):
            break
        contador_pares += 1
        acumulador_pares += numero

    promedio = determinar_promedio(contador_pares, acumulador_pares)
    if promedio is not None:
        print(f"El promedio de los numeros ingresados es: {promedio}")
    else:
        print("No se puede calcular el promedio! No hay numeros pares")
```

```python
if __name__ == "__main__":
    main()
```

```python
"""
Ejercicio Nro 13
Realizar un programa que determine el máximo de una serie de números enteros
ingresados por el usuario.
El ingreso finaliza a pedido del usuario.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def determinar_mayor(numero_actual: int, mayor: int) -> int:
    if numero_actual > mayor:
        return numero_actual
    return mayor

def main() -> None:

    mayor = None

    while True:
        numero = ingresar_numero()

        # Si es el primer número ingresado, lo tomamos como el mayor inicial.
        if mayor is None:
            mayor = numero
        else:
         # Comparamos con el mayor actual y actualizamos si es necesario.
            mayor = determinar_mayor(numero, mayor)

        if input("Desea ingresar otro numero S/N: ").upper() != 'S':
            break

    print(f"El mayor es: {mayor}")


if __name__ == "__main__":
    main()
```

```python
"""
Ejercicio Nro 14
Realizar un programa donde se ingrese el nombre de un producto
y el precio determine los precios máximo
y mínimo de los productos ingresados.
Considere que el ingreso de datos finaliza
a petición del usuario si ingresa ‘s’ es un si y si ingresa ‘n’ es no.
"""

# En Python, tanto las tuplas como las listas usan corchetes []
# para acceder a sus elementos por índice.
```

```python
# Pero la tupla se define con paréntesis ()
# o simplemente con comas al retornar (a, b).
# Las listas se definen con corchetes [].
# Por ejemplo: ("Pan", 120.5) es tupla, ["Pan", 120.5] es lista.
# Aunque use corchetes para acceder (ej: x[1]),
# eso no significa que sea una lista.

# En Python, las tuplas se definen con () y son inmutables.
# Las listas se definen con [] y son mutables.
# Si devuelvo varios valores con "return a, b", es una tupla por defecto.
# Para confirmar el tipo de un objeto en tiempo de ejecución, uso type(objeto)

def ingresar_producto() -> tuple[str, float]:
    while True:
        try:
            producto = input("Ingrese nombre de producto: ")
            precio = float(input("Ingrese precio del producto: "))
            return producto, precio
        except ValueError:
            print("Ingrese valores validos!")

def determinar_precio_maximo(actual: tuple[str, float], maximo: tuple[str, float]) ->
tuple[str, int]:
    if actual[1] > maximo[1]:
        return actual
    return maximo

def determina_precio_minimo(actual: tuple[str, float], minimo: tuple[str, float]) ->
tuple[str, int]:
    if actual[1] < minimo[1]:
        return actual
    return minimo

def main() -> None:
    #Ingresamos el primer producto fuera del ciclo

    producto_actual = ingresar_producto()
    producto_max = producto_actual
    producto_min = producto_actual

    while True:
      respuesta = input("Desea ingresar otro producto (S/N): ").upper()
      if respuesta != 'S':
          break

      producto_actual = ingresar_producto()
      producto_max = determinar_precio_maximo(producto_actual, producto_max)
      producto_min = determina_precio_minimo(producto_actual, producto_min)

    print(f"Prod con precio maximo: {producto_max[0]}, ${producto_max[1]}")
    print(f"Prod con precio minimo: {producto_min[0]}, ${producto_min[1]}")

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 15
```

```
Considerando que el producto entero dos números enteros positivos,
a y b, puede expresarse como la suma sucesiva
de a tantas veces como indique b,
diseñe un algoritmo que
calcule el producto entre a y b mediante sumas sucesivas.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Ingrese valores validos!")

def producto(a: int, b: int) -> int:
    suma = 0
    for _ in range(0, b):
        suma+= a
    return suma

def main() -> None:
    numero_a = ingresar_numero()
    numero_b = ingresar_numero()


    print(f"{numero_a} * {numero_b} es: {producto(numero_a, numero_b)}")

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 16
Realizar un programa que permita ingresar números repetidamente,
determinar el máximo de esos valores.
El algoritmo termina cuando se ingresa un 10,
utilice para terminar la repetición una interrupción con break.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def determinar_maximo(actual: int, maximo: int) -> int:
    if actual > maximo:
        return actual
    return maximo

def main() -> None:
    numero = ingresar_numero()
    maximo = numero

    while True:
        numero = ingresar_numero()
        if numero == 10:
```

```
            break
        maximo = determinar_maximo(numero, maximo)

    print(f"El numero maximo es: {maximo}")
if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 17
Tomando el algoritmo del punto 8 para determinar
si un número es primo,
se ingresan repetidamente números,
lo que se pide es que determinar la cantidad de números primos
que fueron ingresados y la suma de los números que no fueron primos.
El ingreso de datos finaliza cuando la cantidad de números primos
sea superior a 7.
"""

def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese numero: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def es_primo(numero: int) -> bool:
    if numero < 2:
        return False

    # alternativa eficiente:
    # for i in range(2, int(numero ** 0.5) + 1):

    for i in range(2, (int)(numero/2) + 1):
        if numero % i == 0:
            return False
    return True

def main() -> None:
    contador_primos = 0
    suma_no_primos = 0

    # Se corta cuando la cantidad de primos es superior a 7

    while contador_primos <= 7:
        numero = ingresar_numero()

        if es_primo(numero):
            contador_primos += 1
        else:
            suma_no_primos += numero

    print(f"Cantidad primos: {contador_primos}. Suma de no primos: {suma_no_primos}")

if __name__ == "__main__":
    main()
```

```python
"""
Ejercicio Nro 18
Realizar un programa que permita ingresar 6 números,
contar la cantidad de números negativos y cero,
y acumular los números positivos.
Mostrar el promedio de los números positivos
y también mostrar la cantidad de negativos iguales a -5.
"""
def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def es_numero_negativo(numero: int) -> bool:
    return numero < 0

def es_numero_cero(numero: int) -> bool:
    return numero == 0

def es_numero_positivo(numero: int) -> bool:
    return numero > 0

def calcular_promedio(suma: int, cantidad: int) -> float:
   if cantidad == 0:
       return 0.0
   return suma / cantidad

def main() -> None:
    veces = 6
    contador_negativos = 0
    cont_neg_igual_a_menos_cinco = 0
    contador_ceros = 0
    suma_positivos = 0
    contador_positivos = 0

    for _ in range(veces):
        numero = ingresar_numero()
        if es_numero_negativo(numero):
            contador_negativos += 1
            if numero == -5:
                cont_neg_igual_a_menos_cinco += 1
        elif es_numero_cero(numero):
                contador_ceros += 1
        elif es_numero_positivo(numero):
            suma_positivos += numero
            contador_positivos += 1

    promedio_positivos = calcular_promedio(suma_positivos, contador_positivos)

    print(f"\nCantidad de números negativos: {contador_negativos}")
    print(f"Cantidad de ceros: {contador_ceros}")
    print(f"Suma de positivos: {suma_positivos}")
    print(f"Promedio de positivos: {promedio_positivos:.2f}")
    print(f"Cantidad de negativos iguales a -5: {cont_neg_igual_a_menos_cinco}")
```

```python
if __name__ == "__main__":
    main()
```

```python
"""
Ejercicio Nro 19
Realizar un programa que permita ingresar números,
hasta que el número ingresado sea impar y este entre 1 y 10.
Determinar el promedio de los números ingresados
que no estuvieron entre 1 y 10.
"""
def ingresar_numero() -> int:
    while True:
        try:
            return int(input("Ingrese un numero: "))
        except ValueError:
            print("Debe ingresar un valor valido!")

def es_impar(numero: int) -> bool:
    return numero % 2 != 0

def calcular_promedio(suma: int, cantidad: int) -> float:
    if cantidad == 0:
        return 0.0
    return suma/cantidad

def main() -> None:

    contador = 0
    suma = 0

    while True:
        numero = ingresar_numero()

        if es_impar(numero) and (numero >= 1 and numero <= 10):
            break

        if not (numero >= 1 and numero <= 10):
            contador += 1
            suma += numero

    promedio = calcular_promedio(suma, contador)
    print(f"Promedio de los números fuera del rango 1-10: {promedio:.2f}")

if __name__ == "__main__":
    main()
```

```python
"""
Ejercicio Nro 20
Realizar un programa que acepte entrada de números.
A cada número extraiga los dígitos,
y luego reescriba el número original con la siguiente regla:
Número    Letra a usar
1           I
2           Z
3           E
4           A
5           S
```

```
6           G
7           T
8           B
9           g
0           O
Ejemplos:
Entrada    Salida
1211          IZII
795            TgS
8879          BBTg
El programa termina cuando se ingresa
un número que forma la palabra BEAST.
Nota: Trabaje el número como una cadena al extraer sus dígitos.
"""

def ingresar_numero() -> str:
    while True:
        # Solicita al usuario que ingrese un número
        # y se asegura de que sea válido (solo dígitos).
        try:
            entrada = input("Ingrese numero: ")
            if entrada.isdigit(): # Verifica que todos los caracteres sean dígitos (0–9)
                return entrada # Devuelve el número como cadena de texto
        except ValueError:
            print("Debe ingresar un numero valido!")

def reescribir_numero(numero: str) -> str:
    conversion = {
        '1': 'I',
        '2': 'Z',
        '3': 'E',
        '4': 'A',
        '5': 'S',
        '6': 'G',
        '7': 'T',
        '8': 'B',
        '9': 'g',
        '0': 'O'
    }
    palabra = "" # Acumulador para construir la palabra final


    # Recorremos cada carácter (dígito) de la cadena
    # Usamos .get(digito, '') para obtener el valor del diccionario:
    # Si el dígito está en el diccionario, devuelve la letra correspondiente
    # Si por alguna razón no está, devuelve una cadena vacía para evitar errores
    for digito in numero:
       # Dame la letra asociada a digito. Si no existe, devolvé una cadena vacía.
        palabra += conversion.get(digito, '')

    return palabra

def main() -> None:
    while True:
        numero = ingresar_numero()  # Pide número como string
        palabra = reescribir_numero(numero)  # Traduce el número a palabra
        print(f"Traducción: {palabra}")
```

```
        # Si la palabra generada es "BEAST", se termina el programa
        if palabra == "BEAST":
            print("¡Se ingresó la palabra BEAST! Fin del programa.")
            break

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 20
Realizar un programa que acepte entrada de números.
A cada número extraiga los dígitos,
y luego reescriba el número original con la siguiente regla:
Número    Letra a usar
1           I
2           Z
3           E
4           A
5           S
6           G
7           T
8           B
9           g
0           O
Ejemplos:
Entrada    Salida
1211          IZII
795            TgS
8879          BBTg
El programa termina cuando se ingresa
un número que forma la palabra BEAST.
Nota: Trabaje el número como una cadena al extraer sus dígitos.
"""

def ingresar_numero() -> int:
    # Pide al usuario ingresar un número entero positivo
    while True:
        try:
            numero = int(input("Ingrese número: "))
            if numero >= 0:
                return numero
            else:
                print("Por favor, ingrese un número positivo.")
        except ValueError:
            print("Debe ingresar un número válido!")

def reescribir_digito(digito: int) -> str:
    # Mapea cada dígito a su correspondiente letra
    conversion = {
        1: 'I',
        2: 'Z',
        3: 'E',
        4: 'A',
        5: 'S',
        6: 'G',
        7: 'T',
        8: 'B',
```

```
        9: 'g',
        0: 'O'
    }
    return conversion.get(digito, '')

def convertir_a_palabra(numero: int) -> str:
    # Extrae dígitos desde el final y construye la palabra
    palabra = ""
    if numero == 0:
        return reescribir_digito(0)

    while numero > 0:
        digito = numero % 10             # Último dígito
        letra = reescribir_digito(digito)
        palabra = letra + palabra        # Agrega la letra al inicio
        numero //= 10                    # Elimina el último dígito

    return palabra

def main() -> None:
    while True:
        numero = ingresar_numero()
        palabra = convertir_a_palabra(numero)
        print(f"Traducción: {palabra}")

        if palabra == "BEAST":
            print("¡Se ingresó la palabra BEAST! Fin del programa.")
            break

if __name__ == "__main__":
    main()
```

```
"""
Ejercicio Nro 21
Realizar un programa que dibuje un triángulo invertido
con altura mínima de 5 líneas.
La salida debería tener la forma:
Altura = 5
    +****+
    +***+
    +**+
    ++
    +
Altura = 10
    +*********+
    +********+
    +*******+
    +******+
    +*****+
    +****+
    +***+
    +**+
    ++
    +
"""

def ingresar_numero() -> int:
```

```python
    while True:
        try:
            entrada = int(input("Ingrese numero (minimo 5): "))
            if entrada < 5:
                print("La entrada debe ser mayor o igual 5")
            else:
                return entrada
        except ValueError:
            print("Debe ingresar un valor valido!")

def dibujar_triangulo(altura: int) -> None:
    print(f"Altura = {altura}")
    for i in range(altura, 0, -1):  # Desde altura hasta 1 (decreciendo)
        linea = "+" + ("*" * i) + "+"
        print(linea)

def main() -> None:
    altura = ingresar_numero()
    dibujar_triangulo(altura)

if __name__ == "__main__":
    main()
```