



Practica 07 - Bucle While y For



Instrucciones

Entregar los ejercicios en uno o varios [archivos.py](#) o en un notebook de Jupyter ([.ipynb](#))

Bucle While

Ejercicio 1: Cuenta regresiva

Crear un programa que imprima una cuenta regresiva desde 10 hasta 1, y al final imprima "¡Despegue!"

Salida esperada:

```
10  
9  
8
```

```
...
2
1
¡Despegue!
```

Ejercicio 2: Tabla de multiplicar

Solicitar un número al usuario y mostrar su tabla de multiplicar del 1 al 10 usando while

Salida esperada (para número 5):

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
...
5 x 10 = 50
```

Ejercicio 3: Suma hasta límite

Sumar números consecutivos (1, 2, 3, ...) hasta que la suma supere 100. Mostrar cuántos números se sumaron y la suma final.

Salida esperada:

```
Se sumaron 14 números
La suma total es: 105
```

Ejercicio 4: Suma interactiva

Crear un programa que solicite números al usuario y los sume. El programa debe detenerse cuando el usuario ingrese 0, y mostrar la suma total.

Ejercicio 5: Juego de adivinanza.

Implementar un juego de adivinanza donde el programa piense un número del 1 al 50 y el usuario tenga que adivinarlo. Dar pistas de "mayor" o "menor".

Ejercicio 6: Factorial.

Crear un programa que calcule el factorial de un número ingresado por el usuario usando un bucle while.

Ejercicio 7: Factorial.

Las siguientes preguntas requieren que expliques el comportamiento de cada código e identifiques la salida . Lo ideal es identificar la salida "a ojo" y luego verificar tu suposición ejecutando el código.

7.1

```
count = 1
while count <= 10:
    if count % 3 == 0:
        count += 2
        continue
    print(count, end=" ")
    count += 1
```

7.2

```
count = 0
while count < 5:
    print(count, end=" ")
    if count == 2:
        break
    count += 1
```

7.3

```
num = 10
while num > 0:
    if num % 2 == 0:
        num -= 1
    else:
```

```
num += 1  
print(num, end=" ")
```

7.4

```
x = 10  
while x == 10 or x == 8 or x == 6:  
    print("Kookie", end = " ")  
    x -= 4  
print() # Salto de línea
```

Ejercicio 8: Hucha.

Escriba un programa que simule una hucha. El programa solicitará primero una cantidad, que será la cantidad de dinero que queremos ahorrar. A continuación, el programa solicitará una y otra vez las cantidades que se irán ahorrando, hasta que el total ahorrado iguale o supere al objetivo.

Ejercicio 9: Sumatoria

Escriba un programa que lea los números enteros desde teclado, hasta que el usuario ingrese el 0. Finalmente, mostrar la sumatoria de todos los números ingresados.

Ejercicio 10. El mayor

Escriba un programa que lea los números enteros positivos de teclado, hasta que el usuario ingrese el 0. Informar cuál fue el mayor número ingresado.

Bucle For

Ejercicio Resuelto 11: Sistema de Análisis de Calificaciones Estudiantiles



Explicar código paso a paso. Mejorarlo y modificarlo a tu gusto.

Enunciado

La Universidad TechPython necesita un sistema para analizar las calificaciones de sus estudiantes. Como programador junior, debes crear un programa que procese una lista de calificaciones y genere un reporte estadístico completo.

Contexto

- Las calificaciones están en una escala de 0 a 100 puntos
- Una calificación es **aprobatoria** si es mayor o igual a 80 puntos
- Una calificación es **perfecta** si es exactamente 100 puntos
- Se considera "**excelencia académica**" si el promedio del grupo es mayor a 90

Datos de entrada

```
calificaciones = [85, 92, 78, 96, 88, 45, 91, 87, 100, 83, 7  
7, 94]
```

Requerimientos del programa

Tu programa debe calcular y mostrar:

1. Estadísticas básicas:

- Número total de estudiantes
- Suma total de todas las calificaciones
- Promedio general del grupo

2. Análisis de aprobación:

- Cantidad de estudiantes aprobados (≥ 80)
- Cantidad de estudiantes reprobados (< 80)
- Porcentaje de aprobación

3. Detección de casos especiales:

- Si existe al menos una calificación perfecta (100)
- Si hay calificaciones muy bajas (< 60)

- Si el grupo alcanza "excelencia académica" (promedio > 90)

4. Clasificación por rangos:

- **Excelente** (90-100): contar cuántos
- **Bueno** (80-89): contar cuántos
- **Regular** (70-79): contar cuántos
- **Deficiente** (60-69): contar cuántos
- **Muy deficiente** (<60): contar cuántos

Restricciones técnicas

- Debes usar **exactamente un bucle** para procesar todas las calificaciones
- Implementa los tres patrones: **contador**, **acumulador** y **testigo**
- No uses funciones built-in como `sum()`, `max()`, `min()` o `len()` para los cálculos principales
- El programa debe ser robusto y manejar listas vacías

Formato de salida esperado

```
==== REPORTE DE CALIFICACIONES ===
```

ESTADÍSTICAS BÁSICAS:

- Total de estudiantes: 12
- Suma total de calificaciones: 1020
- Promedio general: 85.00

ANÁLISIS DE APROBACIÓN:

- Estudiantes aprobados: 8
- Estudiantes reprobados: 4
- Porcentaje de aprobación: 66.67%

CASOS ESPECIALES:

- ¿Hay calificación perfecta?: Sí

- ¿Hay calificaciones muy bajas?: Sí
- ¿Excelencia académica?: No

DISTRIBUCIÓN POR RANGOS:

- Excelente (90-100): 4 estudiantes
- Bueno (80-89): 4 estudiantes
- Regular (70-79): 2 estudiantes
- Deficiente (60-69): 0 estudiantes
- Muy deficiente (<60): 2 estudiantes

Desafío adicional

Modifica tu programa para que también identifique:

- La posición (índice) donde se encuentra la primera calificación perfecta
- Si hay más de 3 estudiantes en situación crítica (<60)
- El rango de calificaciones más común

Solución

```
# Solución: Sistema de Análisis de Calificaciones Estudiantiles

# Datos de entrada
calificaciones = [85, 92, 78, 96, 88, 45, 91, 87, 100, 83, 77, 94]

# Validar que la lista no esté vacía
if not calificaciones:
    print("Error: No hay calificaciones para procesar")
else:
    # === INICIALIZACIÓN DE VARIABLES ===

    # CONTADORES
```

```

total_estudiantes = 0
contador_aprobados = 0
contador_reprobados = 0
contador_excelente = 0      # 90-100
contador_bueno = 0          # 80-89
contador_regular = 0        # 70-79
contador_deficiente = 0     # 60-69
contador_muy_deficiente = 0 # <60
contador_criticos = 0       # <60 (para desafío adicional)

# ACUMULADORES
acumulador_suma = 0

# TESTIGOS (FLAGS)
testigo_calificacion_perfecta = False
testigo_calificaciones_bajas = False
testigo_excelencia_academica = False
testigo_muchos_criticos = False # Para desafío adicional

# Variables para desafío adicional
posicion_primera_perfecta = -1
indice_actual = 0

# === PROCESAMIENTO CON UN SOLO BUCLE ===
for calificacion in calificaciones:
    # CONTADOR: Total de estudiantes
    total_estudiantes += 1

    # ACUMULADOR: Suma total
    acumulador_suma += calificacion

    # TESTIGO: Calificación perfecta
    if calificacion == 100:
        testigo_calificacion_perfecta = True
        # Desafío adicional: posición de la primera perfe
cta

```

```

        if posicion_primera_perfecta == -1:
            posicion_primera_perfecta = indice_actual

# TESTIGO: Calificaciones muy bajas
if calificacion < 60:
    testigo_calificaciones_bajas = True
    contador_criticos += 1

# CONTADORES: Aprobados vs Reprobados
if calificacion >= 80:
    contador_aprobados += 1
else:
    contador_reprobados += 1

# CONTADORES: Clasificación por rangos
if 90 <= calificacion <= 100:
    contador_excelente += 1
elif 80 <= calificacion <= 89:
    contador_bueno += 1
elif 70 <= calificacion <= 79:
    contador_regular += 1
elif 60 <= calificacion <= 69:
    contador_deficiente += 1
else: # < 60
    contador_muy_deficiente += 1

indice_actual += 1

# === CÁLCULOS POSTERIORES AL BUCLE ===

# Promedio
promedio = acumulador_suma / total_estudiantes

# Porcentaje de aprobación
porcentaje_aprobacion = (contador_aprobados / total_estudiantes) * 100

```

```

# TESTIGO: Excelencia académica
if promedio > 90:
    testigo_excelencia_academica = True

# TESTIGO: Muchos estudiantes críticos (desafío adicional)
if contador_criticos > 3:
    testigo_muchos_criticos = True

# Encontrar el rango más común (desafío adicional)
rangos = {
    "Excelente": contador_excelente,
    "Bueno": contador_bueno,
    "Regular": contador_regular,
    "Deficiente": contador_deficiente,
    "Muy deficiente": contador_muy_deficiente
}

rango_mas_comun = ""
max_estudiantes = 0
for rango, cantidad in rangos.items():
    if cantidad > max_estudiantes:
        max_estudiantes = cantidad
        rango_mas_comun = rango

# === PRESENTACIÓN DE RESULTADOS ===

print("==> REPORTE DE CALIFICACIONES ==>")
print()

print("ESTADÍSTICAS BÁSICAS:")
print(f"- Total de estudiantes: {total_estudiantes}")
print(f"- Suma total de calificaciones: {acumulador_suma}")
print(f"- Promedio general: {promedio:.2f}")

```

```

print()

print("ANÁLISIS DE APROBACIÓN:")
print(f"- Estudiantes aprobados: {contador_aprobados}")
print(f"- Estudiantes reprobados: {contador_reprobados}")
print(f"- Porcentaje de aprobación: {porcentaje_aprobacion:.2f}%")
print()

print("CASOS ESPECIALES:")
print(f"- ¿Hay calificación perfecta?: {'Sí' if testigo_calificacion_perfecta else 'No'}")
print(f"- ¿Hay calificaciones muy bajas?: {'Sí' if testigo_calificaciones_bajas else 'No'}")
print(f"- ¿Excelencia académica?: {'Sí' if testigo_excelencia_academica else 'No'}")
print()

print("DISTRIBUCIÓN POR RANGOS:")
print(f"- Excelente (90-100): {contador_excelente} estudiantes")
print(f"- Bueno (80-89): {contador_bueno} estudiantes")
print(f"- Regular (70-79): {contador_regular} estudiantes")
print(f"- Deficiente (60-69): {contador_deficiente} estudiantes")
print(f"- Muy deficiente (<60): {contador_muy_deficiente} estudiantes")
print()

# === DESAFÍO ADICIONAL ===

print("ANÁLISIS ADICIONAL:")
if testigo_calificacion_perfecta:
    print(f"- Primera calificación perfecta en posición: {posicion_primera_perfecta}")

```

```

else:
    print("- No hay calificaciones perfectas")

    print(f"- ¿Más de 3 estudiantes en situación crítica?:"
{'Sí' if testigo_muchos_criticos else 'No'}")
    print(f"- Estudiantes en situación crítica (<60): {contad
or_criticos}")
    print(f"- Rango más común: {rango_mas_comun} ({max_estudi
antes} estudiantes)")
    print()

# === RESUMEN DE PATRONES UTILIZADOS ===

print("== PATRONES IMPLEMENTADOS ==")
print("CONTADORES utilizados:")
    print(" • total_estudiantes, contador_aprobados, contado
r_reprobados")
    print(" • contador_excelente, contador_bueno, contador_r
egular, etc.")
    print(" • contador_criticos")
print()
print("ACUMULADORES utilizados:")
    print(" • acumulador_suma (para calcular el promedio)")
print()
print("TESTIGOS utilizados:")
    print(" • testigo_calificacion_perfecta")
    print(" • testigo_calificaciones_bajas")
    print(" • testigo_excelencia_academica")
    print(" • testigo_muchos_criticos")

```

Ejercicio 12: Sistema de Análisis de Ventas Mensuales

Enunciado

La empresa "ElectroTech" necesita un sistema para analizar las ventas de sus productos durante el último año. Como desarrollador de sistemas, debes crear un programa que procese las ventas mensuales y genere un informe detallado para la gerencia.

Contexto

- Las ventas están expresadas en miles de dólares
- Una venta es considerada "**exitosa**" si supera los \$50,000 (50 en la lista)
- Una venta es "**excepcional**" si supera los \$80,000 (80 en la lista)
- Se considera "**año próspero**" si el promedio mensual supera los \$60,000 (60 en la lista)
- Un mes es "**crítico**" si las ventas son menores a \$30,000 (30 en la lista)

Datos de entrada

```
ventas_mensuales = [45, 62, 38, 71, 55, 89, 23, 76, 82, 41, 67, 58]  
# Representa las ventas de enero a diciembre (en miles de dólares)
```

Requerimientos del programa

Tu programa debe calcular y mostrar:

1. Estadísticas generales:

- Número total de meses analizados
- Ventas totales del año
- Promedio mensual de ventas
- Identificar el mes con mayor venta y su valor

2. Análisis de rendimiento:

- Cantidad de meses con ventas exitosas ($\geq \$50k$)

- Cantidad de meses con ventas regulares ($< \$50k$)
- Porcentaje de meses exitosos
- Cantidad de meses excepcionales ($\geq \$80k$)

3. Detección de situaciones especiales:

- Si hubo al menos un mes excepcional
- Si hubo meses críticos ($< \$30k$)
- Si fue un "año próspero" (promedio $> \$60k$)
- Si hubo más de 2 meses consecutivos por debajo de $\$40k$

4. Clasificación trimestral:

- **Q1** (Ene-Mar): promedio del primer trimestre
- **Q2** (Abr-Jun): promedio del segundo trimestre
- **Q3** (Jul-Sep): promedio del tercer trimestre
- **Q4** (Oct-Dic): promedio del cuarto trimestre
- Identificar cuál fue el mejor trimestre

5. Análisis de tendencias:

- Contar cuántos meses tuvieron crecimiento respecto al mes anterior
- Contar cuántos meses tuvieron decrecimiento
- Detectar si terminó el año mejor que como empezó (dic $>$ ene)

Restricciones técnicas

- Debes usar **exactamente un bucle** para procesar todas las ventas mensuales
- Implementa los tres patrones: **contador, acumulador y testigo**
- No uses funciones built-in como `sum()`, `max()`, `min()` para los cálculos principales
- El programa debe manejar correctamente listas vacías
- Para los promedios trimestrales, haz los cálculos después del bucle principal

Formato de salida esperado

==== INFORME ANUAL DE VENTAS ELECTROTECH ===

ESTADÍSTICAS GENERALES:

- Meses analizados: 12
- Ventas totales del año: \$667,000
- Promedio mensual: \$55,583
- Mes con mayor venta: Mes 6 con \$89,000

ANÁLISIS DE RENDIMIENTO:

- Meses exitosos ($\geq \$50k$): 7
- Meses regulares ($< \$50k$): 5
- Porcentaje de éxito: 58.33%
- Meses excepcionales ($\geq \$80k$): 2

SITUACIONES ESPECIALES:

- ¿Hubo mes excepcional?: Sí
- ¿Hubo meses críticos?: Sí
- ¿Año próspero?: No
- ¿Más de 2 meses consecutivos bajos?: Sí

ANÁLISIS TRIMESTRAL:

- Q1 (Ene-Mar): \$48,333
- Q2 (Abr-Jun): \$71,667
- Q3 (Jul-Sep): \$60,333
- Q4 (Oct-Dic): \$55,333
- Mejor trimestre: Q2

TENDENCIAS:

- Meses con crecimiento: 6
- Meses con decrecimiento: 5
- ¿Terminó mejor que empezó?: Sí

Desafío adicional

Implementa también:

- Identificar el trimestre con mayor variabilidad (diferencia entre el mes más alto y más bajo del trimestre)
- Detectar si hubo una "racha ganadora" de 3 o más meses consecutivos exitosos
- Calcular en qué posición (mes) ocurrió la primera venta excepcional
- Determinar si la segunda mitad del año fue mejor que la primera mitad

Pistas para la solución

- Usa una variable para llevar el índice actual durante el bucle
- Para detectar meses consecutivos, compara con variables que guarden el estado anterior
- Los cálculos trimestrales se pueden hacer después del bucle principal usando los datos ya procesados
- Recuerda inicializar correctamente todas las variables antes del bucle

Ejercicio 13. Imprime desde el último

Escribir un programa que pida al usuario una palabra y luego muestre por pantalla una a una las letras de la palabra introducida empezando por la última.

Ejercicio 14. Suma y promedio

Escribir un programa que permita al usuario ingresar 6 números enteros, que pueden ser positivos o negativos. Al finalizar, mostrar la sumatoria de los números negativos y el promedio de los positivos. No olvides que no es posible dividir por cero, por lo que es necesario evitar que el programa arroje un error si no se ingresaron números positivos.

Ejercicio 15. Múltiplos de 3

Escribir un programa que muestre la sumatoria de todos los múltiplos de 3 encontrados entre el 0 y el 100.

Ejercicio 16. Divisores

Escriba un programa que le pida al usuario un número entero mayor que cero y que luego imprima en pantalla sus divisores.

Ejercicio 17. De tres en tres

Imprimir los números entre el 5 y el 20, saltando de tres en tres