Programación para Física y Astronomía

Departamento de Física.

Coordinadora: C Loyola
Profesores C Femenías / F Bugini / D Basantes

Primer Semestre 2025

Universidad Andrés Bello
Departamento de Física y Astronomía

Universidad Andrés Bello®

python powered

Resumen - Semana 2, Sesión 1 (Sesión 3)

Introducción a la Sesión 3

Estructuras de Control: Condicionales

Estructuras de Control: Bucles

Ejercicios Guiados

Actividad Práctica

Consolidación y Retroalimentación

Conclusiones

Introducción a la Sesión 3

- **Introducir** las estructuras de control fundamentales en Python.
- **Dominar** el uso de condicionales (`if`, `elif`, `else`).
- **Comprender** bucles básicos (`for`, `while`) y su aplicación.
- **Aplicar** estas estructuras en problemas físicos y matemáticos.
- **Desarrollar** lógica de programación a través de ejercicios prácticos.

Introducción a la Sesión 3 ∋ Conexión con Sesiones Previas

- Sesión 1 y 2 (Semana 1) cubrieron:
  - Introducción al entorno Google Colab.
  - Variables, tipos de datos (`int`, `float`, `str`).
  - Operaciones aritméticas y entrada/salida (`input()`, `print()`).
  - Ejercicios simples de asignación y cálculos básicos.
- Ahora daremos el **salto crucial**: hacer que nuestros programas puedan **tomar decisiones** y **repetir acciones**.

Estructuras de Control: Condicionales

Estructuras de Control: Condicionales ∋ ¿Qué son las Estructuras Condicionales?

- Hasta ahora, nuestros programas ejecutan todas las líneas en **secuencia**.
- Las **estructuras condicionales** permiten que el programa tome **decisiones**.
- Ejecutan diferentes bloques de código según se cumplan ciertas **condiciones**.

Analogía Física

Como un fotón que toma diferentes caminos según su energía: si $E > E_{umbral}$ → efecto fotoeléctrico, sino → no hay emisión.

Sintaxis fundamental:

```
if condición:
    # Código que se ejecuta SI la condición es True
    instrucción1
    instrucción2
```

Puntos clave:

- La `condición` debe evaluarse como `True` o `False`.
- Los **dos puntos (:)** son obligatorios.
- La **indentación** (4 espacios) define qué código está dentro del `if`.

Estructuras de Control: Condicionales ∋ Ejemplo Práctico: Clasificación de Temperatura

```python
temperatura = float(input("Temperatura del agua (°C): "))

if temperatura > 100:
    print("El agua está en estado gaseoso (vapor)")

if temperatura >= 0 and temperatura <= 100:
    print("El agua está en estado líquido")

if temperatura < 0:
    print("El agua está en estado sólido (hielo)")
```

Operadores de comparación:

- `>`, `<`, `>=`, `<=`, `==`, `!=`
- `and`, `or`, `not` para combinar condiciones

Estructuras de Control: Condicionales ∋ La Estructura `if-elif-else` Completa

Sintaxis mejorada para múltiples condiciones:

```
if condición1:
    # Se ejecuta si condición1 es True
    código1
elif condición2:
    # Se ejecuta si condición1 es False y condición2 es True
    código2
elif condición3:
    # Se ejecuta si las anteriores son False y condición3 es
    ↪ True
    código3
else:
    # Se ejecuta si TODAS las condiciones anteriores son False
    código_por_defecto
```

Importante: Solo se ejecuta **UNO** de los bloques (el primero que sea `True`).

Clasificación de Temperatura

```python
temperatura = float(input("Temperatura del agua (°C): "))

if temperatura > 100:
    estado = "gaseoso (vapor)"
    energia = "alta"
elif temperatura >= 0:
    estado = "líquido"
    energia = "media"
else:  # temperatura < 0
    estado = "sólido (hielo)"
    energia = "baja"

print(f"A {temperatura}°C, el agua está en estado {estado}")
print(f"Nivel de energía cinética molecular: {energia}")
```

Ventaja: Más eficiente y lógico que múltiples `if` independientes.

Estructuras de Control: Bucles

- Los **bucles** permiten repetir un bloque de código múltiples veces.
- Evitan la necesidad de escribir el mismo código una y otra vez.
- Python tiene dos tipos principales: `for` y `while`.

Analogía Física

Como las órbitas planetarias: el planeta repite su trayectoria alrededor del sol hasta que alguna fuerza externa (condición) cambie el sistema.

Sintaxis:

```
while condición:
    # Código que se repite MIENTRAS la condición sea True
    instrucción1
    instrucción2
    # IMPORTANTE: modificar algo para que eventualmente sea
    ↪  False
```

Características:

- Se repite **mientras** la condición sea `True`.
- Si la condición nunca se vuelve `False` → **bucle infinito**.
- Útil cuando **no sabemos** cuántas repeticiones necesitamos.

Estructuras de Control: Bucles ∋ Ejemplo: Aproximación a la Raíz Cuadrada (Método de Newton)

```python
numero = float(input("Número para calcular raíz cuadrada: "))
aproximacion = numero / 2  # Estimación inicial
tolerancia = 0.0001

print(f"Calculando raíz cuadrada de {numero}...")

while abs(aproximacion**2 - numero) > tolerancia:
    aproximacion = (aproximacion + numero/aproximacion) / 2
    print(f"Aproximación actual: {aproximacion}")

print(f"Raíz cuadrada ≈ {aproximacion}")
print(f"Verificación: {aproximacion}² = {aproximacion**2}")
```

Física relevante: Métodos iterativos son fundamentales en simulaciones físicas.

Sintaxis:

```
for variable in range(inicio, fin, paso):
    # Código que se repite para cada valor de 'variable'
    instrucción1
    instrucción2
```

Ejemplos de range():

- `range(5)` → 0, 1, 2, 3, 4
- `range(1, 6)` → 1, 2, 3, 4, 5
- `range(0, 10, 2)` → 0, 2, 4, 6, 8
- `range(10, 0, -1)` → 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

Útil cuando: Sabemos exactamente cuántas repeticiones necesitamos.

```python
print("Tabla de Conversión Celsius → Fahrenheit")
print("="*40)

for celsius in range(0, 101, 10):  # De 0°C a 100°C, cada 10°
    fahrenheit = (9/5) * celsius + 32
    print(f"{celsius:3d}°C = {fahrenheit:5.1f}°F")

print("="*40)
```

Salida esperada:

```
  0°C =  32.0°F
 10°C =  50.0°F
 20°C =  68.0°F
 ...
100°C = 212.0°F
```

Ejercicios Guiados

Ejercicios Guiados ∋ Ejercicio 1: Clasificador de Velocidades

Enunciado

- Pedir al usuario la velocidad de un objeto (m/s).
- Clasificar según rangos físicos:
  - $v < 1$ m/s: "Movimiento lento"
  - $1 \leq v < 10$ m/s: "Movimiento moderado"
  - $10 \leq v < 100$ m/s: "Movimiento rápido"
  - $v \geq 100$ m/s: "Movimiento muy rápido"
- Mostrar también la energía cinética si se proporciona la masa.

Conceptos: Uso de `if-elif-else` y cálculos físicos.
Física relevante: Escalas de velocidad en diferentes contextos físicos.

Ejercicios Guiados ∋ Ejercicio 2:
Suma de Números Pares

Enunciado

- Calcular la suma de todos los números pares desde 2 hasta un número n dado por el usuario.
- Usar un bucle `for` con `range()`.
- Mostrar cada número par que se suma y el total final.
- Verificar el resultado usando la fórmula: $\sum_{k=1}^{n/2} 2k = n(n/2+1)$ para n par.

Conceptos: Bucles `for`, acumuladores, validación matemática.
Física relevante: Sumas de series son comunes en física estadística.

Ejercicios Guiados ∋ Ejercicio 3: Juego de Adivinanza con Física

Enunciado

- El programa ”piensa” en la velocidad de la luz en el vacío (299,792,458 m/s).
- El usuario debe adivinar este número.
- Usar un bucle `while` que continúe hasta que el usuario acierte.
- Dar pistas: ”muy alto”, ”alto”, ”bajo”, ”muy bajo” según la proximidad.
- Contar el número de intentos.

Conceptos: Bucles `while`, condicionales, contadores.
Física relevante: Constantes físicas fundamentales.

Ejercicios Guiados ∋ Solución 1 de Referencia: Clasificador de Velocidades ✓

```python
# Entrada de datos
velocidad = float(input("Velocidad del objeto (m/s): "))
masa = float(input("Masa del objeto (kg, opcional, 0 si no aplica): "))
# Clasificación de velocidad
if velocidad < 1:
    categoria = "Movimiento lento"
elif velocidad < 10:
    categoria = "Movimiento moderado"
elif velocidad < 100:
    categoria = "Movimiento rápido"
else:
    categoria = "Movimiento muy rápido"

print(f"Velocidad: {velocidad} m/s → {categoria}")
# Cálculo de energía cinética si se proporciona masa
if masa > 0:
    energia_cinetica = 0.5 * masa * velocidad**2
    print(f"Energía cinética: {energia_cinetica} J")
```

Ejercicios Guiados ∋ Solución 2 de Referencia: Suma de Números Pares ✓

```python
# Entrada de datos
n = int(input("Ingrese el número límite: "))

# Inicializar acumulador
suma_pares = 0

print(f"Números pares desde 2 hasta {n}:")

# Bucle para sumar números pares
for numero in range(2, n + 1, 2):  # Desde 2, hasta n+1, de 2 en 2
    suma_pares += numero
    print(f"  + {numero}")

print(f"Suma total de números pares: {suma_pares}")

# Verificación con fórmula (solo si n es par)
if n % 2 == 0:
    formula_resultado = n * (n // 2 + 1)
```

Ejercicios Guiados ∋ Solución 3 de Referencia: Juego de Adivinanza con Física ✓

```python
# Número secreto: velocidad de la luz en m/s
numero_secreto = 299792458
intentos = 0

print("Adivina la velocidad de la luz en el vacío (m/s)")
print("Pista: es un número de 9 dígitos")

while True:
    intento = int(input("Tu estimación: "))
    intentos += 1

    diferencia = abs(intento - numero_secreto)

    if intento == numero_secreto:
        print(f"¡Correcto! La velocidad de la luz es
        ↪  {numero_secreto} m/s")
        print(f"Lo lograste en {intentos} intentos")
        break
    elif diferencia < 1000000:  # Dentro de 1 millón
```

Actividad Práctica

Actividad Práctica ∋ Organización para Trabajo en Equipos

Formación de equipos

- Grupos de 2-3 personas
- Cada persona crea su propio notebook
- Comparten pantalla y discuten soluciones

Metodología de trabajo

- **10 min**: Lean y discutan los problemas
- **20 min**: Implementen las soluciones
- **5 min**: Prueben con diferentes valores
- **10 min**: Comparen resultados entre compañeros

Actividad Práctica ∋ Actividad Extra 1: Calculadora de Período Orbital

Enunciado

- Calcular el período orbital de planetas usando la 3ª Ley de Kepler: $T^2 = \frac{4\pi^2}{GM}a^3$
- Pedir el semieje mayor a en metros.
- Usar $G = 6.674 \times 10^{-11}$ m³/(kg·s²) y $M_{sol} = 1.989 \times 10^{30}$ kg.
- Mostrar el período en segundos, días y años.
- Usar condicionales para clasificar el tipo de órbita.

Objetivo: Integrar estructuras de control con cálculos astrofísicos.

Actividad Práctica ∋ Actividad Extra 2: Simulador de Decaimiento Radioactivo

Enunciado

- Simular el decaimiento radioactivo usando: $N(t) = N_0 e^{-\lambda t}$
- Pedir: número inicial de átomos N_0 y vida media $t_{1/2}$.
- Calcular $\lambda = \frac{\ln(2)}{t_{1/2}}$.
- Usar un bucle `for` para mostrar $N(t)$ cada 100 años hasta 1000 años.
- Agregar condicionales para alertar cuando quede menos del 10% y 1%.

Objetivo: Bucles con aplicaciones en física nuclear.

Consolidación y Retroalimentación

Cada equipo comparta brevemente:

- ¿Qué estructura de control les resultó más difícil de entender?
- ¿Encontraron errores comunes? ¿Cómo los solucionaron?
- ¿Algún truco o descubrimiento interesante con bucles o condicionales?
- ¿Cómo validaron que sus cálculos físicos fueran correctos?

Objetivo

Aprender de las experiencias de otros equipos y normalizar que los errores son parte del aprendizaje.

Consolidación y Retroalimentación ∋ Errores Comunes en Estructuras de Control

Errores frecuentes con IF

- Olvidar los dos puntos (:) después de la condición
- Problemas de indentación (usar tabs y espacios mezclados)
- Usar = (asignación) en lugar de == (comparación)

Errores frecuentes con bucles

- **Bucle infinito**: No modificar la variable de control en `while`
- **Off-by-one**: Confusión con los límites de `range()`
- **Indentación**: No alinear correctamente el código dentro del bucle

Consejo

Usar `print()` para debuggear: imprimir variables dentro de bucles para entender qué está pasando.

Conclusiones

Conclusiones ∋ Resumen de la Sesión

- Dominamos las **estructuras condicionales** (`if`, `elif`, `else`).
- Aprendimos sobre **bucles** (`while`, `for`) y sus aplicaciones.
- Aplicamos estas estructuras a **problemas físicos reales**.
- Desarrollamos **lógica de programación** a través de ejercicios progresivos.
- Practicamos **debugging** y resolución colaborativa de problemas.

Habilidad adquirida

Sus programas ahora pueden tomar decisiones inteligentes y automatizar tareas repetitivas.

- **Sesión 4 (Semana 2)**: Introducción a **funciones** en Python.
- **Temas futuros**:
  - Definición de funciones y parámetros
  - Reutilización de código y modularización
  - Funciones en bibliotecas científicas
- **Práctica recomendada**: Crear pequeños programas que combinen condicionales y bucles.

Próxima sesión

Aprenderán a organizar su código en **funciones reutilizables** para resolver problemas más complejos.

- **Python Docs - Control Flow** (documentación oficial).
- **Real Python - Conditionals** (tutoriales avanzados).
- **Real Python - For Loops** (ejemplos prácticos).
- **LearnPython.org - Loops** (ejercicios interactivos).

Práctica adicional

Prueben modificar los ejercicios de hoy agregando más condicionales o cambiando los rangos de los bucles.

Entrega (Canvas)

- Sube tu notebook con los 3 ejercicios principales resueltos
- Incluye comentarios explicando tu razonamiento
- Al menos una de las actividades extra completada
- Fecha límite: antes de la próxima clase

Autoevaluación

- ¿Entiendo cuándo usar `if-elif-else` vs múltiples `if`?
- ¿Sé cuándo elegir `for` vs `while`?
- ¿Puedo debuggear bucles infinitos?

¡Excelente progreso!

- Ahora dominan las **estructuras de control fundamentales**
- Sus programas pueden **tomar decisiones** y **automatizar tareas**
- Están listos para **funciones** y **modularización**
- Recuerden: **la práctica constante** es clave

¡Nos vemos en la Sesión 4!