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 2 (Sesión 4)

Introducción y Conexión

Ejercicios Prácticos Integrados

Trabajo Colaborativo

Soluciones de Referencia

Análisis y Actividades Extra

Evaluación y Retroalimentación

Conclusiones

Introducción y Conexión

- **Sesión 3 (Semana 2-1)** dominamos:
  - Estructuras de control fundamentales (if, elif, else).
  - Bucles básicos (for, while) y su aplicación.
  - Condicionales y toma de decisiones en programas.
  - Ejercicios prácticos con aplicaciones físicas.
- **Objetivo de hoy**: Aplicar y consolidar estas estructuras de control en problemas más elaborados y colaborativos.

- **Consolidar** el uso de estructuras de control (if, elif, else, for, while).
- **Aplicar** estas estructuras en problemas físicos y matemáticos complejos.
- **Desarrollar** habilidades de resolución colaborativa de problemas.
- **Integrar** múltiples conceptos de programación en aplicaciones prácticas.
- **Preparar** el camino hacia funciones y modularización de código.

Ejercicios Prácticos Integrados

- **No más teoría**: Ya dominamos if, elif, else, for, while.
- **Enfoque 100% práctico**: Resolver problemas físicos complejos.
- **Integración de conceptos**: Combinar múltiples estructuras de control.
- **Trabajo colaborativo**: Equipos para resolver desafíos progresivos.

Meta de la Sesión

Que cada estudiante se sienta cómodo aplicando estructuras de control en problemas reales de física y matemáticas.

Ejercicios Prácticos Integrados ∋ Actividad Central: Problemas Paso a Paso

- Realizaremos 5 **ejercicios progresivos** que integran:
  - Estructuras condicionales complejas (if-elif-else)
  - Bucles de repetición avanzados (for, while)
  - Aplicaciones en contexto físico real
  - Validación de datos y manejo de errores
- Cada ejercicio se abordará primero en **colaboración** y luego se compartirá la solución.
- Objetivo: Consolidar el uso de estructuras de control en problemas reales.

Ejercicios Prácticos Integrados ∋ Ejercicio 1: Ecuación de Movimiento en 1D

Enunciado

- Dados los siguientes parámetros físicos:
  - `x0`: posición inicial (m)
  - `v0`: velocidad inicial (m/s)
  - `a`: aceleración constante (m/s²)
  - `t`: tiempo (s)
- Calcular la posición final usando la ecuación cinemática:

$$x(t) = x_0 + v_0 \cdot t + \frac{1}{2}at^2$$

- Mostrar el resultado con unidades apropiadas.

Conceptos: Ecuaciones cinemáticas, variables de entrada, cálculos secuenciales.
Física relevante: Movimiento rectilíneo uniformemente acelerado (MRUA).

Ejercicios Prácticos Integrados ∋ Ejercicio 2: Promedio y Varianza de Mediciones

Enunciado

- Solicitar al usuario **3 mediciones** físicas (pueden ser temperaturas, distancias, etc.).
- Utilizar un bucle `for` para recopilar los datos.
- Calcular el **promedio** ($\bar{x}$) y la **varianza** muestral.
- Fórmula de varianza muestral:

$$s^2 = \frac{\sum_{i=1}^{3}(x_i - \bar{x})^2}{n-1}$$

- Mostrar ambos resultados con formato apropiado.

Conceptos: Bucles `for`, listas, acumuladores, estadística básica.
Física relevante: Análisis estadístico de mediciones experimentales.

Ejercicios Prácticos Integrados ∋ Ejercicio 3: Conversión de Unidades con Condicionales

Enunciado

- Crear un programa que solicite al usuario:
  - Un valor numérico
  - Una unidad origen: `"cm"`, `"m"`, `"km"`
  - Una unidad destino: `"cm"`, `"m"`, `"km"`
- Usar estructuras `if-elif-else` para determinar la conversión.
- Calcular y mostrar el resultado con las unidades correspondientes.
- Manejar casos de unidades inválidas con mensajes de error.

Conceptos: Condicionales múltiples, validación de entrada, factores de conversión.
Física relevante: Sistema métrico de unidades, conversiones de longitud.

Ejercicios Prácticos Integrados ∋ Ejercicio 4: Tabla de Multiplicar con Bucles

Enunciado

- Solicitar al usuario un número entero.
- Usar un bucle `for` para generar la tabla de multiplicar de ese número.
- Mostrar los resultados del 1 al 10 en formato: `"N x i = resultado"`.
- Agregar una validación para verificar que el número ingresado sea positivo.

Conceptos: Bucles `for`, validación con `if`, formato de salida.
Física relevante: Relaciones proporcionales, escalado de magnitudes.

Ejercicios Prácticos Integrados ∋ Ejercicio 5: Clasificador de Temperaturas

Enunciado

- Solicitar al usuario una lista de temperaturas en grados Celsius.
- Usar un bucle `for` para procesar cada temperatura.
- Clasificar cada temperatura usando `if-elif-else`:
  - Menor a 0°C: "Congelación"
  - 0°C a 25°C: "Frío"
  - 25°C a 35°C: "Templado"
  - Mayor a 35°C: "Calor"
- Mostrar un resumen final con la cantidad de temperaturas en cada categoría.

Conceptos: Bucles, condicionales anidados, contadores, procesamiento de listas.
Física relevante: Estados de la materia, escalas de temperatura.

Trabajo Colaborativo

- Formar **grupos de 2-3 estudiantes**.
- Seleccionar 2-3 ejercicios de los 5 propuestos (según el tiempo disponible).
- Editar un **notebook compartido** en Google Colab.
- **Estrategia recomendada**:
  - Ejercicios 1-2: Fundamentales (ecuaciones, estadística)
  - Ejercicios 3-4: Intermedios (condicionales, validaciones)
  - Ejercicio 5: Avanzado (integración de conceptos)
- **Objetivo**: Discutir soluciones, anotar dudas y resolver en conjunto.

- ¿Cuál de los ejercicios fue el más complejo?
- ¿En qué parte surgieron errores recurrentes?
- ¿Cómo podría hacerse un **diseño modular** (dividir el problema en funciones)?
- ¿Qué estrategias de debugging utilizaron?

Comparte tus experiencias con la clase.

Soluciones de Referencia

Soluciones de Referencia ∋ Solución 1 de Referencia: Ecuación de Movimiento en 1D ✓

```python
# Solicitar datos al usuario con unidades claras
x0 = float(input("Posición inicial x0 (m): "))
v0 = float(input("Velocidad inicial v0 (m/s): "))
a = float(input("Aceleración a (m/s²): "))
t = float(input("Tiempo t (s): "))

# Aplicar la ecuación cinemática
x_final = x0 + v0 * t + 0.5 * a * (t**2)

# Mostrar resultado con formato claro
print(f"La posición final es: {x_final:.2f} m")
```

Discusión: Uso directo de float() para simplificar entrada, formato de decimales en salida.

Soluciones de Referencia ∋ Solución 2 de Referencia: Promedio y Varianza de Mediciones ✓

```python
# Recopilar datos usando bucle for
mediciones = []
for i in range(1, 4):
    valor = float(input(f"Ingrese medición {i}: "))
    mediciones.append(valor)

# Calcular promedio
promedio = sum(mediciones) / len(mediciones)

# Calcular varianza muestral
suma_diferencias = 0
for valor in mediciones:
    suma_diferencias += (valor - promedio)**2

varianza = suma_diferencias / (len(mediciones) - 1)

# Mostrar resultados
print(f"Promedio: {promedio:.3f}")
print(f"Varianza muestral: {varianza:.3f}")
```

Discusión: Uso de len() para generalización, acumulador para varianza, formato de decimales.

Soluciones de Referencia ∋ Solución 3 de Referencia: Conversión de Unidades con Condicionales ✓

```python
# Solicitar datos al usuario
valor = float(input("Ingrese el valor numérico: "))
unidad_origen = input("Unidad origen (cm, m, km): ").lower()
unidad_destino = input("Unidad destino (cm, m, km): ").lower()

# Convertir primero todo a metros (unidad base)
if unidad_origen == "cm":
    valor_metros = valor / 100
elif unidad_origen == "m":
    valor_metros = valor
elif unidad_origen == "km":
    valor_metros = valor * 1000
else:
    print("Unidad de origen no válida")
    valor_metros = None

# Convertir de metros a unidad destino
if valor_metros is not None:
    if unidad_destino == "cm":
        resultado = valor_metros * 100
    elif unidad_destino == "m":
        resultado = valor_metros
    elif unidad_destino == "km":
        resultado = valor_metros / 1000
    else:
        print("Unidad de destino no válida")
        resultado = None

    if resultado is not None:
        print(f"Resultado: {resultado:.3f} {unidad_destino}")
```

Soluciones de Referencia ∋ Solución 4 de Referencia: Tabla de Multiplicar con Bucles ✓

```python
# Solicitar número al usuario
numero = int(input("Ingrese un número para su tabla de multiplicar: "))

# Validar que sea positivo
if numero > 0:
    print(f"Tabla de multiplicar del {numero}:")
    print("-" * 25)

    # Generar tabla usando bucle for
    for i in range(1, 11):
        resultado = numero * i
        print(f"{numero} x {i} = {resultado}")
else:
    print("Por favor, ingrese un número positivo.")
```

Discusión: Validación con if, bucle for con range, formato de salida organizado.

Soluciones de Referencia ∋ Solución 5 de Referencia: Clasificador de Temperaturas ✓

```python
# Solicitar temperaturas
temperaturas = []
num_temp = int(input("¿Cuántas temperaturas desea clasificar? "))

for i in range(num_temp):
    temp = float(input(f"Temperatura {i+1} (°C): "))
    temperaturas.append(temp)

# Contadores para cada categoría
congelacion = frio = templado = calor = 0

# Clasificar cada temperatura
for temp in temperaturas:
    if temp < 0:
        print(f"{temp}°C: Congelación")
        congelacion += 1
    elif temp <= 25:
        print(f"{temp}°C: Frío")
        frio += 1
    elif temp <= 35:
        print(f"{temp}°C: Templado")
        templado += 1
    else:
        print(f"{temp}°C: Calor")
        calor += 1

# Resumen final
print(f"\nResumen: Congelación={congelacion}, Frío={frio}, "
      f"Templado={templado}, Calor={calor}")
```

Análisis y Actividades Extra

- Estructuras de control integradas:
  - `for` loops con `range()`: Ejercicios 2, 4 y 5
  - `if-elif-else`: Ejercicios 3 y 5
  - Validaciones: Ejercicios 4 y 5
- Conceptos de programación aplicados:
  - Acumuladores: varianza (Ej. 2), contadores (Ej. 5)
  - Listas dinámicas: `append()` en múltiples ejercicios
  - Formato de salida: `f-strings` con decimales
- Buenas prácticas observadas:
  - Comentarios explicativos en cada sección
  - Validación de entrada de usuario
  - Nombres de variables descriptivos

Análisis y Actividades Extra ∋ Actividad Extra 1: Calculadora Físico-Química Interactiva

Enunciado

- Crear un programa que presente un menú al usuario con las opciones:
  - (1) Calcular densidad: $\rho = \frac{m}{V}$ (kg/m³)
  - (2) Calcular fuerza: $F = m \cdot a$ (N)
  - (3) Calcular energía cinética: $E_c = \frac{1}{2}mv^2$ (J)
  - (4) Salir del programa
- Usar `if-elif-else` para procesar la selección del usuario.
- Solicitar los datos necesarios y mostrar el resultado con unidades.
- Permitir al usuario realizar múltiples cálculos hasta que elija salir.

Objetivo: Integrar menús, condicionales, bucles y cálculos físicos en una aplicación completa.

Simulador de Caída Libre

Enunciado

- Simular la caída libre de un objeto usando: $y(t) = y_0 - \frac{1}{2}gt^2$
- Pedir altura inicial y_0 y gravedad g (9.81 m/s² por defecto).
- Usar un bucle `while` para calcular la posición cada 0.1 segundos.
- Detenerse cuando el objeto toque el suelo ($y \leq 0$).
- Mostrar tabla con tiempo, altura y velocidad en cada paso.
- Calcular tiempo total de caída y velocidad final.

Objetivo: Bucles `while` con condiciones físicas y simulación temporal.

Evaluación y Retroalimentación

Evaluación y Retroalimentación ∋ Tarea Semanal: Entrega en Canvas

Entrega Obligatoria

- Completar los 5 ejercicios principales con código funcional.
- Incluir comentarios explicando el razonamiento en cada paso.
- Al menos una de las actividades extra completada.
- Documentar cualquier dificultad encontrada y cómo se resolvió.

Criterios de Evaluación

- Código funcional y sin errores (40%)
- Uso correcto de estructuras de control (30%)
- Comentarios y documentación (20%)
- Actividad extra y creatividad (10%)

- ¿Alguno de los ejercicios resultó especialmente difícil?
- ¿Cómo han manejado los **mensajes de error** (entradas inválidas)?
- ¿Qué estrategias de debugging resultaron más útiles?
- ¿Se sienten preparados para abordar funciones en la próxima unidad?

Importante

La colaboración es clave, pero cada estudiante debe entender completamente su código antes de entregarlo.

Conclusiones

Conclusiones ∋ Síntesis de la Sesión 4

- **Consolidamos** el uso práctico de estructuras de control:
  - Condicionales complejas (if-elif-else)
  - Bucles aplicados (for, while)
  - Validación de datos y manejo de errores
- **Aplicamos** programación a problemas físicos reales:
  - Ecuaciones cinemáticas y análisis estadístico
  - Conversiones de unidades y procesamiento de datos
  - Simulaciones y clasificaciones automáticas
- **Desarrollamos** habilidades de trabajo colaborativo y debugging.

- **Próximamente**: **Unidad IV** - Funciones en Python
- **Temas por venir**:
  - Definición de funciones con `def`
  - Parámetros, argumentos y valores de retorno
  - Modularización de código y reutilización
  - Funciones en bibliotecas científicas (NumPy, Matplotlib)
- **Preparación recomendada**:
  - Identificar código repetitivo en los ejercicios de hoy
  - Pensar en cómo dividir problemas complejos en partes más simples
  - Practicar la documentación de código con comentarios claros

- **Python Docs - Control Flow** (referencia oficial).
- **LearnPython.org** (ejercicios interactivos).
- **Real Python - Python Basics** (tutoriales profundos).
- **Real Python - Functions** (preparación para próxima unidad).

Práctica Adicional

Intenten modificar los ejercicios de hoy agregando más validaciones o cambiando los parámetros físicos para diferentes escenarios.

- **Practicar diariamente**: Sesiones cortas pero consistentes (20-30 min).
- **Explorar aplicaciones reales**: Usar datos de experimentos, astronomía, física.
- **Documentar y comentar**: El código se olvida rápido sin documentación.
- **Depurar sistemáticamente**: Usar `print()` para entender el flujo.
- **Colaborar y discutir**: Explicar código a otros consolida el aprendizaje.

¡Excelente progreso!

- Ya dominan las **estructuras de control fundamentales**
- Pueden aplicar programación a **problemas físicos reales**
- Están listos para **funciones y modularización**
- Recuerden: **la práctica constante es clave**

¡Nos vemos en la próxima unidad: Funciones!