Programación para Física y Astronomía

Departamento de Física.

Corodinadora: C Loyola
Profesoras/es C Loyola / C Femenías / Y Navarrete / C Ruiz / F Bugini

Primer Semestre 2025

Universidad Andrés Bello
Departamento de Física y Astronomía

Universidad Andrés Bello®

python powered

Resumen - Semana 5, Sesión 1 (Sesión 9)

Introducción y Contexto

Objetivos de la Sesión 9

- **Repasar** de forma integral los contenidos vistos en las semanas anteriores.
- **Resolver** ejercicios y dudas previas a la evaluación **Solemne I**.
- **Fortalecer** la comprensión de sintaxis, estructuras de control, funciones y módulos.
- **Identificar** áreas con más dificultades y reforzarlas antes del examen parcial.

Recapitulación de Semanas 1-4

- **Semana 1**: Configuración de Google Colab, operaciones básicas, `input/print`, variables.
- **Semana 2**: Sintaxis, aritmética, estructuras de control iniciales (`if, while`).
- **Semana 3**: Profundización de `while`, introducción al `for`, ejercicios colaborativos.
- **Semana 4**: **Funciones**, **módulos** y uso de **librerías externas** (`pip`, `numpy`, etc.).

Resumen de Contenidos Clave

Sintaxis General y Tipos de Datos

- **Indentación**: bloques de código (`if`, `while`, `for`, `def`).
- **Tipos básicos**: `int`, `float`, `str`, `bool`, `complex`.
- **Operaciones**: `+ - * / // % **`, prioridad de operadores, paréntesis.
- **Manejo de E/S**: `input()`, `print()`, conversión de tipos (`int()`, `float()`).

Condicionales `if/elif/else`

- **if condicion:** bloque si `True`.
- **elif condicion_2:** bloque si `condicion_2` es `True`.
- **else:** bloque final si ninguna condición anterior se cumple.
- **Ejemplos**:
  - Menús interactivos.
  - Validaciones de rango (p.e. notas entre 1.0 y 7.0).

Bucles `while` y `for`

- **while condicion:** repite bloque mientras la condición sea `True`.
  - `break` para salir, `continue` para saltar a la siguiente iteración.
- **for** sobre secuencias (`for i in range(...):`), ideal cuando conocemos la cantidad de iteraciones.
- **Aplicaciones**:
  - Lectura indefinida de datos.
  - Iteración sobre listas y rangos.

Funciones

- **def nombre(parámetros):** define una nueva función.
- **return** para devolver un valor (opcional).
- **Parámetros por defecto** (`def f(x=10): ...`) y **keyword arguments**.
- **Ventajas**: modularidad, reuso, claridad de código.
- **Scope**: variables locales dentro de la función.

Módulos y Paquetes

- **Módulo** = archivo `.py` con funciones, clases, variables reutilizables.
- **Paquete** = carpeta con `__init__.py` y varios módulos.
- Importación mediante `import modulo` o `from modulo import func`.
- **Ejemplos**:
  - `import math`, `import numpy as np`, `import mi_modulo`.

Librerías Externas

- Instalación con `pip install <paquete>` o `!pip install <paquete>` en Colab.
- Ejemplo: `numpy` para cálculo numérico, `matplotlib` para gráficas.
- Importancia de la **colaboración open-source** y la documentación.

Ejercicios de Repaso

Ejercicio 1: Repaso de Fundamentos

Enunciado

- Pide un número entero **n**.
- Calcula su factorial (**n!**) de dos formas:
  1. Con un **for**.
  2. Mediante una **función recursiva**.
- Muestra ambos resultados y valida que coincidan.

Objetivo: Reforzar sintaxis de bucles y definición de funciones.

Ejercicio 2: Análisis de Notas con `while`

Enunciado

- Solicitar notas en un `while` hasta que el usuario ingrese `-1`.
- Validar que cada nota esté en el rango [1.0, 7.0].
- Llevar conteo del número de notas válidas, suma total y promedio.
- Al final, imprimir el promedio o un mensaje si no hay datos válidos.

Objetivo: Revisar **while**, validaciones, conteo y promedio.

Ejercicio 3: Módulo de Conversión de Unidades

Enunciado

- Crea un archivo `conversor.py` con varias funciones:
  - `cm_a_m`, `m_a_km`, `km_a_cm`, etc.
- Importa `conversor` en `main.py` y pide al usuario un valor y la conversión deseada (ej. cm $\rightarrow$ km).
- Muestra el resultado final.

Objetivo: Practicar la creación de **módulos**, importación y lógica simple de funciones.

Ejercicio 4: Usando numpy

Enunciado

- Instala e importa **numpy**.
- Genera un **arreglo** de 10 números aleatorios (`np.random.rand(10)`).
- Calcula su **media** y **desviación estándar**.
- **Opcional**: filtrar solo valores > 0.5 y mostrarlos.

Objetivo: Reforzar **numpy**, acceso a funciones `mean`, `std`, slicing y condicionales.

Actividad Colaborativa

- Trabaja en **parejas** o **grupos de 3**.
- Selecciona 2 ejercicios (o más) y discútelos en conjunto.
- Anota cualquier duda o error que surja durante la implementación.
- Comparte tus soluciones y reflexiona sobre los puntos más complicados.

Ayuda y Orientación

- Recuerda el uso de **docstrings** en funciones para clarificar su propósito.
- Maneja **ValueError** cuando conviertes `input` a `int` o `float`.
- Asegúrate de importar módulos correctamente y de guardar los archivos en la misma carpeta (o configurar la ruta).
- Si usas **numpy** en Colab, revisa si necesitas **!pip install numpy** o si ya está incluido (por defecto Colab lo incluye).

Discusión y Consolidación

Puesta en Común

- ¿Qué ejercicios causaron más dificultad?
- ¿Qué estrategias de resolución fueron más efectivas?
- ¿Dudas persistentes sobre **tipos**, **bucles** o **importaciones**?

Comparte con la clase para beneficio de todos.

Dudas Frecuentes Identificadas

- **Scope de variables** dentro de funciones.
- Mezclar `if` y `while` en un mismo flujo (ej.: menús interactivos).
- **Errores de importación** con rutas inadecuadas.
- **Excepción** al convertir datos de entrada que no son numéricos.

Buenas Prácticas para el Solemne

- **Lee bien el enunciado** y comprende la tarea antes de programar.
- **Pseudocódigo breve**: planifica el flujo de control (if, while, for).
- **Funciones claras**: si el problema lo amerita, divide la lógica en funciones.
- **Pruebas con casos simples**: revisa siempre el comportamiento con inputs distintos.
- **Comentarios y docstrings**: facilitan la compresión de tu solución (y parcial o total puntaje).

Ejemplo de Pregunta: *Función y Validación*

Enunciado

- Define una función `leer_entero_positivo(msg)` que:
  - Muestra el mensaje `msg`.
  - Lee un entero desde `input()`.
  - Valida que sea **positivo**.
  - Si no lo es, vuelve a pedir el número hasta que sea válido.
  - Retorna el valor correcto.
- Usa esta función para leer `n` y luego imprimir la suma de 1 a `n`.

Consejos y Cierre

Consejos de Estudio

- **Revisar apuntes** y ejercicios de las clases pasadas.
- **Practicar** sintaxis de Python en Colab (ej.: if, while, for, funciones).
- **Hacer miniprogramas** que integren varios conceptos (entradas, salidas, validaciones).
- **Releer** la documentación o apuntes de funciones clave (`math`, `random`, `numpy`, etc.).

Autoevaluación: Preguntas Guía

- ¿Puedo leer datos de usuario y convertirlos correctamente?
- ¿Sé escribir un **bucle while** que termina en la condición correcta?
- ¿Entiendo cómo `if/elif/else` deciden el flujo?
- ¿Me siento cómodo definiendo funciones con `def` y `return`?
- ¿Podría crear un módulo `mimodulo.py` y usarlo en un script principal?

Recursos Adicionales

- **Python Tutorial Oficial** (repaso secciones 1-6).
- **W3Schools Python** (ejemplos básicos).
- **Stack Overflow** (buscar soluciones a errores comunes).
- Videos y guías en YouTube: “`Python for Beginners`”, “`Programación en Python`”.

Próxima Sesión: Solemne I

- **No hay contenido nuevo**, sino la **evaluación** de los temas ya vistos:
  - Unidades I y II (Syllabus).
  - Sintaxis básica, control de flujo, funciones y primeros usos de módulos.
- **Formato de la Solemne**: problemas cortos y medianos que requieren programar en Python y quizá un par de preguntas conceptuales.

¡Mucho éxito en la Solemne!

- Aprovechen de **revisar ejercicios**.
- No duden en **consultar** a través de foros o con sus compañeros.
- ¡Nos vemos en la siguiente sesión (Solemne I)!