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 3, Sesión 2 (Sesión 6)

Introducción y Repaso

Módulos y Paquetes

Librerías Externas

Ejercicios Prácticos Integrados

Tarea 2

Conclusiones y Próximos Pasos

Introducción y Repaso

Introducción y Repaso ∋ Recapitulación de la Sesión Anterior (Sesión 6)

- **Semana 3, Sesión 2 (Sesión 6)** se centró en:
  - **Funciones**: sintaxis (`def`), parámetros, valores por defecto, alcance de variables.
  - **Módulos y Paquetes**: cómo organizar el código en archivos `.py` y carpetas.
  - Ejemplos de proyectos pequeños con `import` y definición de funciones útiles.
- **Objetivo de hoy**: Ampliar la práctica con funciones y módulos, e **introducir el uso de librerías externas** (vía `pip` o Colab).

- **Profundizar** en el flujo de trabajo al crear y reutilizar módulos en Python.
- **Explorar** la instalación de librerías externas (`pip`, Google Colab).
- **Diseñar** una actividad grupal donde se combine la creación de funciones propias con el uso de librerías de terceros.
- **Fomentar** la colaboración y la discusión sobre buenas prácticas de organización.

Módulos y Paquetes

- **Carpetas y librerías**: Organiza tu proyecto en carpetas y utiliza librerías conocidas para funcionalidades específicas.
- `main.py`: Archivo principal que orquesta la lógica del programa.
- Ejemplo de estructura:
  - `main.py`: Código principal.
  - `data/`: Carpeta para almacenar datos.
  - `requirements.txt`: Lista de librerías necesarias.
- **Ventaja**: Facilita la mantenibilidad, escalabilidad y reutilización del código.

Módulos y Paquetes ∋ Formas de Importar Librerías Conocidas

Import completo: Útil para acceder a todas las funcionalidades de una librería.

```
1  import numpy as np
2  arr = np.array([1, 2, 3])
3  print(np.mean(arr))
```

From / Import: Importa solo las funciones necesarias.

```
1  from math import sqrt
2  resultado = sqrt(16)
3  print(resultado)
```

Import renombrado: Simplifica el uso de librerías con nombres largos.

```
1  import pandas as pd
2  df = pd.DataFrame({"A": [1, 2], "B": [3, 4]})
3  print(df)
```

Librerías Externas

Librerías Externas ∋ ¿Por qué Librerías Externas?

- **Ahorra tiempo**: aprovechas código ya probado por la comunidad.
- **Funcionalidades avanzadas**: Desde manejo de redes hasta machine learning.
- **Ejemplos**: `requests` para peticiones web, `numpy` para cálculo numérico, `pandas` para data frames, etc.
- **Comunidad activa**: librerías mantenidas, actualizaciones frecuentes.

Librerías Externas ∋ Instalación con `pip`

- **pip**: el gestor de paquetes oficial de Python.
- Comando general en terminal:

```
pip install nombre_paquete
```

- Si usas **Google Colab**, puedes instalar temporalmente en una celda:

```
1  !pip install nombre_paquete
```

- La librería quedará disponible para importarse en el resto del entorno (hasta reiniciar).

Librerías Externas ∋ Ejemplo: `requests` en Colab

```
# En una celda de Colab:
!pip install requests

import requests

resp = requests.get("https://api.github.com")
print(resp.status_code)
print(resp.json())
```

- **Uso real**: Conectarse a APIs, descargar datos, etc.
- **Sugerencia**: Manejar casos de error (`resp.status_code != 200`).

Librerías Externas ∋ Ejemplo: numpy Básico

- Instalación desde terminal local:

```
pip install numpy
```

- Uso en el código:

```python
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr * 2)  # [2 4 6 8]
```

- **NumPy** es la base de muchas librerías científicas en Python.
- **Operaciones vectorizadas**: eficiencia y simplicidad.

Ejercicios Prácticos Integrados

Ejercicios Prácticos Integrados ∋ Ejercicio 1: Análisis de Temperaturas

Enunciado

- Crear una función `clasificar_temperatura(temp)` que clasifique una temperatura en grados Celsius:
  - $< 0\,^{\circ}\mathrm{C}$: "Estado sólido (hielo)"
  - $0\,^{\circ}\mathrm{C} - 100\,^{\circ}\mathrm{C}$: "Estado líquido (agua)"
  - $> 100\,^{\circ}\mathrm{C}$: "Estado gaseoso (vapor)"
- Crear un programa principal (.ipynb de la clase) que:
  - Pida al usuario 3 temperaturas diferentes
  - Use la función para clasificar cada temperatura
  - Muestre el resultado para cada temperatura ingresada

Conceptos: Funciones básicas, condicionales if-elif-else.
Física relevante: Estados de la materia, puntos de cambio de fase.

Ejercicios Prácticos Integrados ∋ Ejercicio 2: Conversor de Unidades Simple

Enunciado

- Crear un archivo `convertidor.py` con dos funciones:
  - `celsius_a_fahrenheit(c)` → retorna $F = C \times 9/5 + 32$
  - `metros_a_pies(m)` → retorna $pies = metros \times 3.281$
- En otro archivo (.ipynb de la clase):
  - Importar las funciones del módulo `convertidor`
  - Pedir al usuario un valor y la conversión deseada (1: C→F, 2: m→pies)
  - Mostrar el resultado de la conversión

Conceptos: Módulos básicos, importación, funciones simples, condicionales.
Física relevante: Sistemas de unidades y conversiones básicas.

Ejercicios Prácticos Integrados ∋ Ejercicio 3: Calculadora de Caída Libre

Enunciado

- Crear una función `posicion_caida(t, h0, g=9.8)`:
  - Calcula posición usando $h = h0 - \frac{1}{2}g \cdot t^2$
  - Parámetros: t (tiempo), h0 (altura inicial), g (gravedad)
- Crear un programa principal (.ipynb de la clase) que:
  - Pida al usuario la altura inicial de un objeto
  - Calcular y mostrar la posición cada segundo (de 0 a 5 segundos)
  - Imprimir "¡El objeto tocó el suelo!" si la posición es menor o igual a 0

Conceptos: Funciones con parámetros por defecto, bucle for simple, condicionales básicos.
Física relevante: Cinemática básica, caída libre.

Ejercicios Prácticos Integrados ∋ Ejercicio 4: Estadísticas Básicas

Enunciado

- Crear un módulo `estadistica.py` con una función:
  - `calcular_promedio(valores)` - retorna la media aritmética
- Crear un programa principal (.ipynb de la clase) que:
  - Importe la función desde el módulo
  - Pida al usuario que ingrese 5 mediciones de temperatura
  - Valide que las entradas sean números (use un bucle while si es necesario)
  - Calcule y muestre el promedio de las mediciones

Conceptos: Módulos básicos, funciones simples, listas, bucle for, validación básica.
Física relevante: Mediciones de temperatura, promedio de datos experimentales.

Ejercicios Prácticos Integrados ∋ Ejercicio 5: Mini-Calculadora Científica

Enunciado

- Crear un módulo `calculadora.py` con dos funciones:
  - `energia_cinetica(masa, velocidad)` - calcula $E_c = \frac{1}{2}mv^2$
  - `energia_potencial(masa, altura, g=9.8)` - calcula $E_p = mgh$
- Crear un programa principal (.ipynb de la clase) que:
  - Importar las funciones del módulo
  - Presentar un menú simple con opciones (1: E. Cinética, 2: E. Potencial)
  - Pedir los datos necesarios según la opción elegida
  - Mostrar el resultado del cálculo

Conceptos: Módulos básicos, funciones, menú simple con if-elif, entrada de usuario.

Física relevante: Energía cinética y potencial, conservación de energía mecánica.

Tarea 2

Tarea 2 ∋ Tarea Pregunta 1: Calculadora de Movimiento Simple

Enunciado

- Definir una función `calcular_posiciones(velocidad_inicial, tiempo_max, intervalo=1)` que:
  - Calcule las posiciones de un objeto en movimiento rectilíneo uniforme
  - La posición se determina por $x(t) = v \cdot t$ (donde v es la velocidad constante)
  - Use un bucle `for` para calcular la posición en cada intervalo de tiempo desde 0 hasta `tiempo_max`
  - Si la posición supera 100 unidades, incluya un mensaje ”Fuera de rango” junto al valor
- Ejemplo de uso:
  - `calcular_posiciones(10, 5)` debería calcular la posición cada segundo durante 5 segundos
  - `calcular_posiciones(15, 10, 0.5)` debería calcular la posición cada medio segundo durante 10 segundos

Tarea 2 ∋ Tarea Pregunta 2: Filtrado y Análisis de Datos Experimentales

Enunciado

- Definir una función `procesar_datos(datos, umbral_min=None, umbral_max=None)` que:
  - Filtre valores atípicos fuera del rango [`umbral_min`, `umbral_max`]
  - Si `umbral_min` es `None`, calcule automáticamente como $media - 2 * desviacion$
  - Si `umbral_max` es `None`, calcule automáticamente como $media + 2 * desviacion$
  - Use `if` para verificar los umbrales
  - Use `for` y `continue` para saltar valores fuera de rango
  - Retorne la lista de valores filtrados

Entrega: Notebook (`.ipynb`) con implementación y pruebas de las funciones.

Física relevante: Tratamiento de datos experimentales, análisis estadístico en mediciones.

Conclusiones y Próximos Pasos

- Ventajas de **combinar módulos propios con librerías externas**:
  - Reutilización de código (módulos).
  - Potencia y robustez (librerías de terceros).
- Importancia de la **organización de archivos** y de un **flujo de trabajo** claro.
- Manejar **entornos virtuales** (en local) es otra buena práctica (tema futuro).

- Revisar la **documentación oficial** de `pip` y `virtualenv`.
- Explorar **PyPI** (`https://pypi.org/`) para descubrir librerías útiles.
- Practicar la creación de **módulos** en proyectos pequeños.
- Investigar qué librerías podrían ser útiles para futuros trabajos de Física/Astronomía (p.e. `astropy`, `scipy`).

- **Sesion 7 (Semana 4)**: Aplicaciones profundas de Unidades I y II.
- **Recomendación**:
  - Revisa todos los conceptos vistos: Sintaxis, Estructuras de Control, Funciones, Módulos.
  - Práctica con ejercicios y ejemplos de exámenes pasados (si los hubiera).

¡Prepárate para la evaluación!

- Official Python Packaging Tutorial
- Documentación de la Biblioteca Estándar de Python
- **PyPI** - Python Package Index
- Numpy Docs
- Matplotlib Docs

¡Muchas gracias y éxito en su práctica!

- Recuerden subir su trabajo a Google Drive o repositorio compartido.
- ¡Sigan explorando librerías externas y creando módulos propios!