Programación para Física y Astronomía

Departamento de Física.

Corodinadora: 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 6, Sesión 2 (Sesión 12)

Introducción y Repaso

Manipulación Avanzada de Arrays

Funciones Random y Linalg

Primeros Pasos con Matplotlib

Ejercicios Prácticos

Conclusiones y Próximos Pasos

Introducción y Repaso

- Vimos la introducción a **NumPy** y creamos arreglos básicos.
- Exploramos:
  - `np.array`, `np.zeros`, `np.arange`.
  - Slicing, indexación y operaciones vectorizadas.
  - Broadcasting con ejemplos simples.
- Resolvimos ejercicios como la aproximación a π con la serie de Leibniz.
- **Objetivo de hoy**: Ampliar funciones avanzadas de NumPy, manipulación más compleja y daremos un primer vistazo a **Matplotlib** para gráficas.

- **Profundizar** en la manipulación de arrays: `reshape`, `transpose`, `concatenate`.
- **Explorar** funciones adicionales de `np.random` y `np.linalg`.
- **Iniciar** la transición a **visualizaciones** básicas con **Matplotlib**.
- **Aplicar** estos conceptos en ejercicios de simulación y análisis de datos sencillos.

Manipulación Avanzada de Arrays

Manipulación Avanzada de Arrays ∈ **reshape** para Cambiar Forma

```python
import numpy as np

arr = np.arange(12)  # [0,1,2,...,11]
print("Arr 1D:", arr)

mat = arr.reshape((3,4))  # 3 filas, 4 columnas
print("Matriz 3x4:\n", mat)
```

- **reshape** no crea una copia, sino una vista (si posible).
- Dimensiones deben **coincidir** en la cantidad total de elementos.

Manipulación Avanzada de Arrays ∈ `transpose` o `.T` para Matrices

```
mat = np.array([[1,2,3],
                [4,5,6]])
print("Original:\n", mat)

transpuesta = mat.T
print("Transpuesta:\n", transpuesta)

# También se puede usar:
# np.transpose(mat)
```

- Cambia ejes (filas pasan a ser columnas y viceversa).

Manipulación Avanzada de Arrays ∈ np.concatenate, np.vstack, np.hstack

```
a = np.array([1,2,3])
b = np.array([4,5,6])

c = np.concatenate((a,b))
print("Concatenate 1D:", c)  # [1 2 3 4 5 6]

mat1 = np.array([[1,2],[3,4]])
mat2 = np.array([[5,6],[7,8]])

res_h = np.hstack((mat1, mat2))
# Horizontal: columnas se suman
print("Horizontal:\n", res_h)
# [[1 2 5 6]
#  [3 4 7 8]]

res_v = np.vstack((mat1, mat2))
# Vertical: filas se suman
print("Vertical:\n", res_v)
# [[1 2]
```

Funciones Random y Linalg

Funciones Random y Linalg ∈ `np.random` para Valores Aleatorios

```python
# Generar arreglo 1D de floats uniformes en [0,1)
rand_floats = np.random.rand(5)
print("Floats aleatorios:", rand_floats)

# Matriz 2x3 de flotantes uniformes
mat_uniform = np.random.rand(2,3)

# Enteros aleatorios en [low, high)
rand_ints = np.random.randint(low=1, high=10, size=5)
print("Enteros aleatorios [1..9]:", rand_ints)

# Fijar semilla para reproducibilidad
np.random.seed(42)
```

- `np.random.randn` para distribución normal estándar.
- **seed** para obtener siempre la misma secuencia.

Funciones Random y Linalg ∈ `np.linalg` para Álgebra Lineal

```python
import numpy as np

A = np.array([[1,2],
              [3,4]])
# Determinante
detA = np.linalg.det(A)

# Inversa
invA = np.linalg.inv(A)

# Autovalores y autovectores
valores, vectores = np.linalg.eig(A)

print("Det(A) =", detA)
print("Inv(A) =\n", invA)
print("Eigenvalues =", valores)
print("Eigenvectors =\n", vectores)
```

Primeros Pasos con Matplotlib

- **Visualizar datos** es fundamental en Física/Astronomía:
  - Gráficas de series temporales, funciones, histogramas, dispersión, etc.
- **Matplotlib** es la librería más común para gráficos 2D en Python.
- **Fácil integración** con NumPy: dibujar arreglos, transformaciones en tiempo real.

```
pip install matplotlib
```

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x = np.linspace(0, 2*np.pi, 100)
5  y = np.sin(x)
6
7  plt.plot(x, y)
8  plt.show()
```

- `pyplot` es la interfaz estilo **MATLAB** para Matplotlib.

```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 2*np.pi, 100)
sinx = np.sin(x)
cosx = np.cos(x)

plt.plot(x, sinx, label='sin(x)')
plt.plot(x, cosx, label='cos(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Ejemplo Seno y Coseno')
plt.legend()
plt.show()
```

- `plt.xlabel`, `plt.ylabel`, `plt.title`, `plt.legend` para etiquetar y personalizar.

```
1  fig, ax = plt.subplots()  # Crea figura y eje
2  ax.plot(x, sinx, 'r-', label="Seno")
3  ax.plot(x, cosx, 'b--', label="Coseno")
4  ax.set_xlabel("Eje X")
5  ax.set_ylabel("Eje Y")
6  ax.set_title("Ejemplo con subplots")
7  ax.legend()
8  plt.show()
```

- Método `subplots` retorna **fig** (figura) y **ax** (eje).
- Permite mayor control al personalizar gráficas.

Ejercicios Prácticos

Ejercicios Prácticos $\in$ Ejercicio 1: Reshape y Concatenación

Enunciado

- Crear un arreglo 1D con los números del 0 al 11 (`np.arange(12)`).
- **Reshape** a una matriz de forma (3,4).
- Crear otra matriz de (3,4) con valores $\{[10, 20, 30, 40], \ldots\}$.
- Concatenar horizontalmente ambas matrices (`np.hstack`) para obtener (3,8).
- Imprimir el resultado final.

Enunciado

- Crear una matriz 3x3 aleatoria (`randint en [1..5]`).
- Calcular su **determinante** y **traer** la **inversa** (si existe).
- Imprimir autovalores y autovectores.
- **Opcional**: revisar casos donde $\det(A) = 0$.

Objetivo: Practicar `np.linalg` y detección de singularidad.

Ejercicios Prácticos ∈ Ejercicio 3: Visualización de Datos Aleatorios

Enunciado

- Generar un **conjunto** de 50 valores `x` y `y` con `np.random.rand(50)`.
- Graficarlos con **Matplotlib** en un diagrama de dispersión (`plt.scatter`).
- Agregar títulos y etiquetas de ejes.
- **Opcional**: colorear los puntos según otro criterio (p.e. `x+y`).

Objetivo: Combinar NumPy (datos aleatorios) con Matplotlib (visualización).

Enunciado

- Definir una **función** `trayectoria_proyectil`($v0, ang$) que retorne los arreglos `x(t)`, `y(t)` para θ y Δt apropiados.
- Graficar la trayectoria en 2D con `plt.plot`.
- Mostrar $\max(y)$ y alcance aproximado.

Sugerencia: Reutilizar $\Delta t = 0.01$ y `np.arange` hasta que `y` ≤ 0.

Ejercicios Prácticos $\in$ Trabajo en Grupos

- Dividir la clase en **pequeños grupos**.
- Seleccionar al menos 2 ejercicios (o todos) para resolver en Colab.
- Discutir estrategias de **NumPy** y **Matplotlib**.
- Al final, hacer un breve **intercambio de soluciones** y observaciones.

- **Planificar** el código: entender la forma de arreglos que necesitas (`reshape`, `ravel`, etc.).
- **Probar** cada paso en celdas separadas, imprimiendo resultados parciales.
- **Agregar** etiquetas y leyendas a los gráficos para mayor claridad.
- Investigar métodos como `np.hsplit`, `np.vsplit`, `np.meshgrid` si tienes tiempo extra.

Ejercicios Prácticos $\in$ Espacio para Dudas

- ¿Algún problema con **reshape** o `transpose`?
- ¿**Concatenate** vs. **hstack/vstack**? Cuándo usar cada uno.
- ¿Dificultades con **linalg** (inversa, determinante, autovalores)?
- ¿Cómo establecer rangos y ejes en Matplotlib?

Conclusiones y Próximos Pasos

- Comparte tus resultados de **tiro parabólico**, **visualización aleatoria**, etc.
- Destaca si usaste **vectorización** para ahorrar bucles.
- ¿Algún reto interesante con `np.linalg`?
- Retroalimentación colectiva: **buenas prácticas** y **errores comunes**.

- **NumPy Avanzado**:
  - Manipulación de forma, transposición, concatenación y aleatorios.
  - Álgebra lineal básica (`np.linalg`).
- **Primeros pasos con Matplotlib** para visualizar datos.
- Unimos **Física y programación** con ejemplos como **tiro parabólico**.
- Miramos la **importancia de la gráfica** para interpretar resultados.

Conclusiones y Próximos Pasos $\in$ Próximos Temas

- **Semana 7**: Profundizar en la creación de **gráficos avanzados** (histogramas, 3D, subplots múltiples).
- Manejo de datos con `pandas` (posiblemente) y su integración con `matplotlib`.
- **Recomendación**:
  - Practicar la **generación de datos** y la **visualización** en conjunto.
  - Reforzar la manipulación de arrays 2D/3D con NumPy.

- **Documentación oficial de NumPy**
- **Matplotlib Docs** (tutorial de pyplot)
- **Scipy Lectures** (capítulo de NumPy y Matplotlib)
- **Foros y comunidad**: Stack Overflow, Reddit `/r/python`.

Gracias y hasta la próxima sesión

- Sigan experimentando con NumPy y Matplotlib.
- Cualquier duda, pregunten en foros o a compañeros.
- ¡Nos vemos en la **Semana 7** para avanzar en visualización y análisis!