Clase 2: NumPy, Pandas, Matplotlib, SciPy y Scikit-learn

Contenido

Introducción a los principales recursos del ecosistema Python:

Array **-** *Colección ordenada/contigua de elementos del mismo tipo*

NumPy **-** Operaciones vectorizadas, álgebra lineal, etc

Pandas **-** Organización y manipulación de datos tabulares (DataFrames y Series)

Matplotlib **-** Exploración y visualización de datos

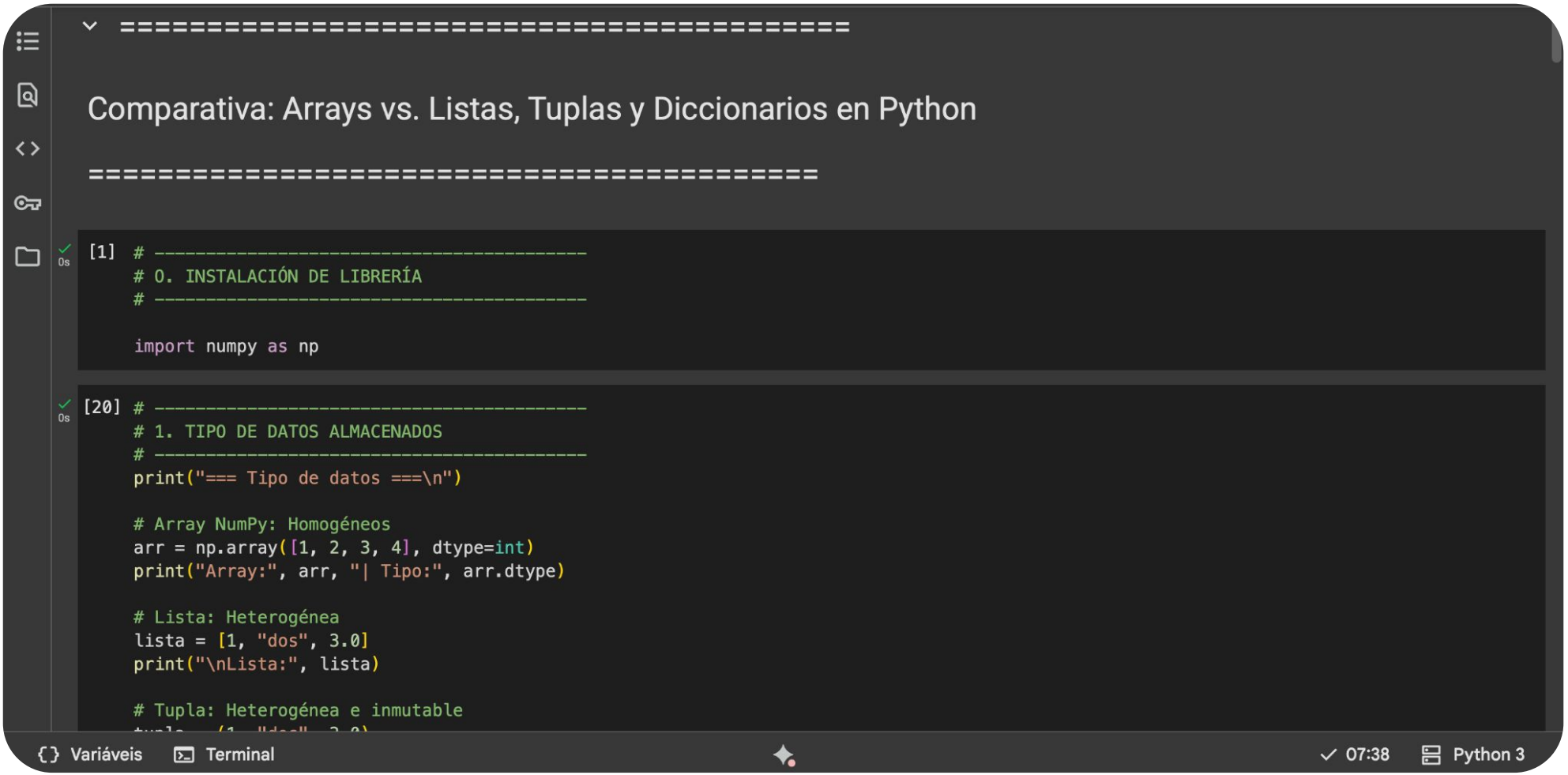
SciPy **-** Análisis: estadística, optimización, transformadas, interpolación, etc

Scikit-learn **-** Aprendizaje de máquina: regresión, clasificación, clustering, etc

NumPy → Pandas → Matplotlib → SciPy → Scikit-learn

Fundamentos Numéricos → Manipulación → Visualización → Análisis Científico → Aprendizaje de Máquina

Notebook - Arrays

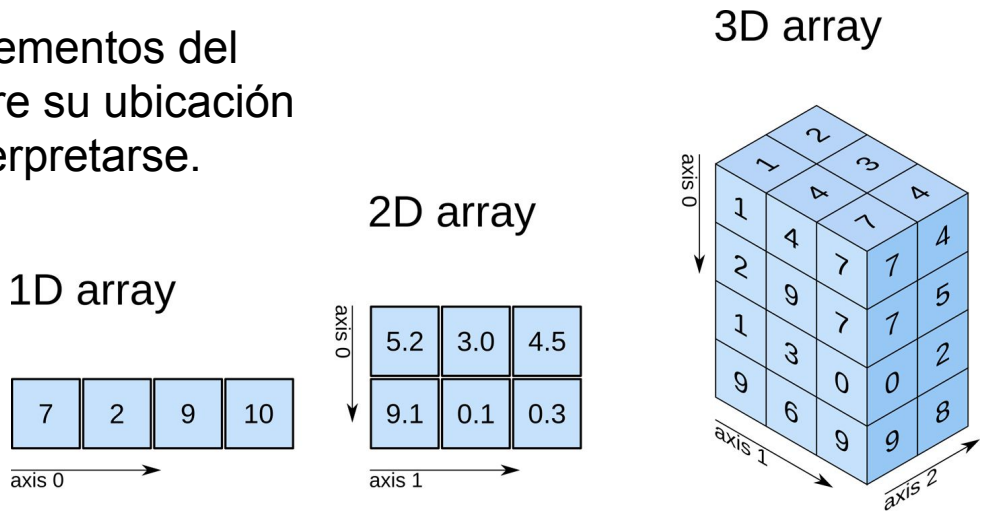


Arrays

Un **array** es una colección ordenada de elementos del mismo tipo, acompañada de información sobre su ubicación en memoria y la forma en que deben interpretarse.

Dónde aparecen en la vida cotidiana:
Imágenes digitales (matriz de píxeles), hojas de cálculo, señales, tablas de datos.

Los ***arrays*** fueron **creados** para resolver problemas prácticos como almacenar y manipular grandes cantidades de datos.



Diferencia con otras estructuras (listas, tuplas, diccionarios)

En lugar de utilizar **múltiples variables individuales**, un array permite agrupar toda la información en una sola estructura, facilitando su **acceso** y **procesamiento**.

Arrays

Arrays vs. Listas, Tuplas y Diccionarios en Python

Tipo de datos almacenados

Arrays → **Homogéneos**: Los elementos son del mismo tipo (enteros o decimales)

Listas → **Heterogéneas**: pueden contener elementos de distintos tipos (ej. enteros, cadenas, objetos)

Tuplas → También **Heterogéneas**, pero **inmutables**: una vez creadas, no se pueden modificar

Diccionarios → Almacenan **pares Clave** (únicas) **– Valor** (cualquier tipo)

Estructura de memoria

Arrays → Memoria contigua → Acceso rápido y eficiente a los elementos

Listas/Tuplas/Diccionarios → Memoria no contigua → Fexibilidad↑ ↓ Eficiencia (Operaciones masivas)

Arrays

Arrays vs. Listas, Tuplas y Diccionarios en Python

Operaciones

Arrays → Soportan operaciones vectorizadas: ideales para cálculo numérico.

Listas/Tuplas → No tienen operaciones vectorizadas; se requiere recorrer elemento por elemento.

Diccionarios → No son para operaciones numéricas masivas, sino para búsquedas rápidas por clave

Mutabilidad

Arrays y **Listas** → Mutables (se pueden modificar sus elementos).

Tuplas → Inmutables.

Diccionarios → Mutables, pero organizados por claves en lugar de índices secuenciales

Arrays

Arrays vs. Listas, Tuplas y Diccionarios en Python

En resumen:

Eficiencia y cálculo numérico masivo → **Arrays** (especialmente con ***NumPy***)

Flexibilidad sin importar la velocidad → **Listas**

Colecciones fijas e inmutables → **Tuplas**

Asociar datos mediante claves → **Diccionarios**

Arrays

Características principales

Índices (acceso posicional)

Homogeneidad (mismo tipo de dato)

Memoria contigua → acceso rápido

Dimensionalidad (1D, 2D, 3D…)

Operaciones básicas

Crear (*zeros*, *ones*, *empty*, arange, linspace, *etc*) / Conocer (*dim*, *shape*, *size*, *type*, *etc*) un ***Array***

Indexación: Acceder a elementos

Ordena de forma creciente (*sort)*

Concatenar/Unir **Arrays** (*concatenate)*

Modificar valores

Recorrer (bucles)

Operaciones en bloque (ej.: sumar todos los elementos)

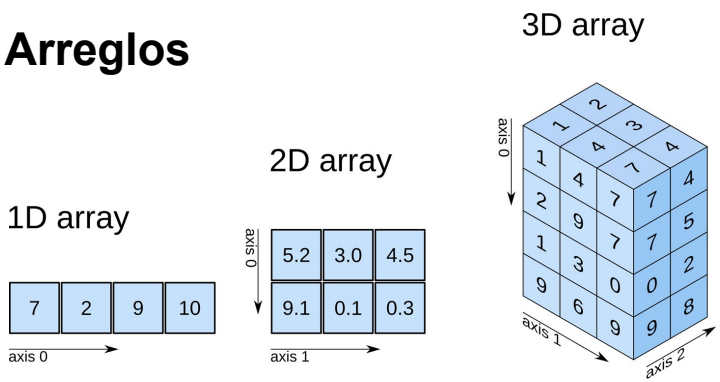
NumPy

import numpy as np ⇨ (alias np: buena práctica)

NumPy (Numerical Python) - Cálculo Numérico y Científico

Arrays n-dimensionales

Arreglos



Los **arrays** de *NumPy* son **arreglos multidimensionales** compactos que facilitan el almacenamiento y procesamiento de grandes volúmenes de datos, siendo considerablemente más rápidos y con menor consumo de memoria que las estructuras nativas de Python, como las listas, lo que los hace ideales para el análisis y modelado de datos.

NumPy proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices) y una variedad de rutinas para operaciones rápidas en matrices, incluyendo **matemáticas, lógica, manipulación de formas, clasificación, selección, *Input/Output*, transformaciones discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más.**

NumPy documentación - Guía para principiantes - Guía de usuários

NumPy

NumPy

El **paquete** fundamental para la computación científica con Python - **Página Oficial**

Fuente abierta: Distribuido bajo una licencia libre de BSD, *NumPy* se desarrolla y mantiene públicamente en GitHub por una comunidad vibrante, receptiva y diversa.

Fácil de usar: La sintaxis de alto nivel de *NumPy* lo hace accesible y productivo para programadores de cualquier origen o nivel de experiencia.

Performante: El núcleo de *NumPy* es un código (C y *Fortran*) bien optimizado. Disfruta de la flexibilidad de Python con la velocidad del código compilado.

Matrices N-dimensionales: Rápidos y versátiles, los conceptos de vectorización, indexación y difusión son los estándares de facto de la computación de matrices hoy en día.

Herramientas de computación numérica: *NumPy* ofrece inúmeras funciones matemáticas integrales, generadores de números aleatorios, rutinas de álgebra lineal, transformaciones de Fourier y más.

Interoperable: Es compatible con una amplia gama de hardware y plataformas informáticas, y funciona bien con librerías distribuidas basadas en GPU (Unidades de Procesamiento Gráfico) y de matriz escasa.

NumPy

Pandas

Historia del desarrollo

Pandas comenzó su desarrollo en **2008** en AQR Capital Management y se convirtió en **código abierto en 2009**, contando desde entonces con una activa comunidad global.
En 2015 pasó a ser patrocinado por NumFOCUS, asegurando su crecimiento como proyecto de primer nivel.

Es una herramienta de análisis y manipulación de datos, rápida, potente, flexible y fácil de usar, construido sobre el lenguaje de programación Python.

⇨ (alias pd: buena práctica)

import pandas as pd

Pandas

Pandas ofrece un **DataFrame rápido y eficiente** con indexación integrada, soporte para **lectura y escritura en múltiples formatos** (CSV, Excel, SQL, HDF5) y **alineación automática de datos** con manejo de valores faltantes.

Permite **reordenar, pivotar, indexar y filtrar** grandes conjuntos de datos, modificar su tamaño, y realizar **operaciones de agrupamiento y agregación** de alto rendimiento. Incluye herramientas para **unión y combinación de datos**, **indexación jerárquica** y amplia **funcionalidad para series temporales**.

Está altamente optimizado (Cython: superset de Python que permite añadir tipado estático y compilación a C para ganar rendimiento) y se usa en diversos campos como finanzas, neurociencia, economía y análisis web.

Pandas

Pandas busca ser el bloque de construcción fundamental de alto nivel para realizar **análisis prácticos** de datos del mundo real en Python. Su objetivo más amplio es convertirse en la herramienta de **análisis y manipulación** de datos de código abierto **más potente y flexible** disponible en cualquier lenguaje.

Un mundo donde el software de análisis y manipulación de datos sea:

- Accesible para todos
- Gratuito para usar y modificar
- Flexible
- Potente
- Fácil de usar
- Rápido

Pandas

Materiales de la página oficial

Primeros pasos: Instalación, Tipo de datos, Leer y escribir datos tabulares, Seleccionar un subconjunto de una tabla, Crear parcelas, Crear nuevas columnas derivadas de las columnas existentes, Calcular las estadísticas resumidas, Remodelar el diseño de las tablas, Combinar datos de varias tablas, Manejar los datos de series temporales y Manipular los datos textuales

Visión general rápida: Estructuras de datos básicas en pandas, Creación de objetos, **Visualización*** y selección de datos, Selección por etiqueta, Selección por posición, Indexación booleana, Configuración, Falta de datos, Operaciones, Estadísticas, Funciones definidas por el usuario, Valores de recuento, Métodos de cadena, Fusionar (Concat, Unir, Agrupación), Reformar, Pila, Tablas dinámicas, Series temporales, Categoricals, Trazado, Importación y exportación de datos (CSV, Parquet, Excel), *Gotchas*.

Hoja de Trucos: Limpieza y preparación de datos (Creación de DataFrames y estructuras básicas, Reordenamiento y transformación de datos, Selección y filtrado de datos, Operaciones con conjuntos, uniones y fusiones, Agrupamiento y operaciones por ventanas, Creación de nuevas columnas y binning, Funciones vectorizadas y transformación de datos, Manejo de datos faltantes, Resumen y estadísticas de datos, Opciones de configuración, Visualización - Plotting, Operaciones sobre cadenas, Conversión y cambio de tipos de datos y Accesores de fecha y hora).

Tutoriales de la comunidad: Orientados principalmente a nuevos usuarios.

Matplotlib

La **librería** *Matplotlib* fue creada por **John D. Hunter en 2003** con el propósito específico de **generar gráficos y visualizaciones de calidad (estáticas, animadas e interactivas) en Python**. La **motivación** detrás de su desarrollo fue crear una herramienta de trazado que pudiera coincidir con las capacidades de trazado que ofrece MATLAB, un software ampliamente utilizado para la computación numérica (Data Science) y la visualización.

Con el propósito de replicar las funcionalidades de MATLAB, **Matplotlib** fue diseñado para ofrecer a los usuarios de Python una solución de trazado familiar (orientada a objeto), potente y flexible, que les permita crear **gráficos de aspecto profesional para varias aplicaciones, incluida la investigación científica, el análisis de datos y la visualización de datos.**

⇨ (alias plt: buena práctica)

import matplotlib.pyplot as plt

paquete general . submódulo que agrupa funciones de trazado

pyplot: (`plot()`, `scatter()`, `imshow()`, etc.)

Matplotlib

Matplotlib facilita lo sencillo y hace posible lo complejo.

> "***Transforma datos aburridos en imágenes cautivadoras, llenando tu mundo de colores, formas y patrones vibrantes***". Bharani K. Depuru

Crear **gráficos y visualizacione de calidad**

Producir **figuras interactivas** que puedan hacer **zoom**, **panorámica**, **actualización**, etc

Personalizar el **estilo visual** y **el diseño**

Exportar a **muchos formatos de archivo**

Incrustar (Conectar) en **JupyterLab e interfaces gráficas de usuario**

Utilizar una amplia gama de **paquetes de terceros**

Empezar - **Ejemplos** - **Referencia** - **Hojas de trucos** - **Documentación**

Matplotlib

¿Qué tipos de gráficos pueden generar los usuarios usando Matplotlib?

Con Matplotlib, los usuarios pueden generar diferentes gráficos como

Tramas de líneas, Gráficos de dispersión/barras/circulares y muchos otros tipos de visualizaciones.

Estos **gráficos y visualizaciones** pueden ayudar a visualizar las relaciones entre variables, analizar patrones en los datos y presentar sus hallazgos a otros. Permite la **personalización de colores, marcadores, estilos de línea, etiquetas y otros atributos visuales** para crear gráficos visualmente atractivos e informativos.

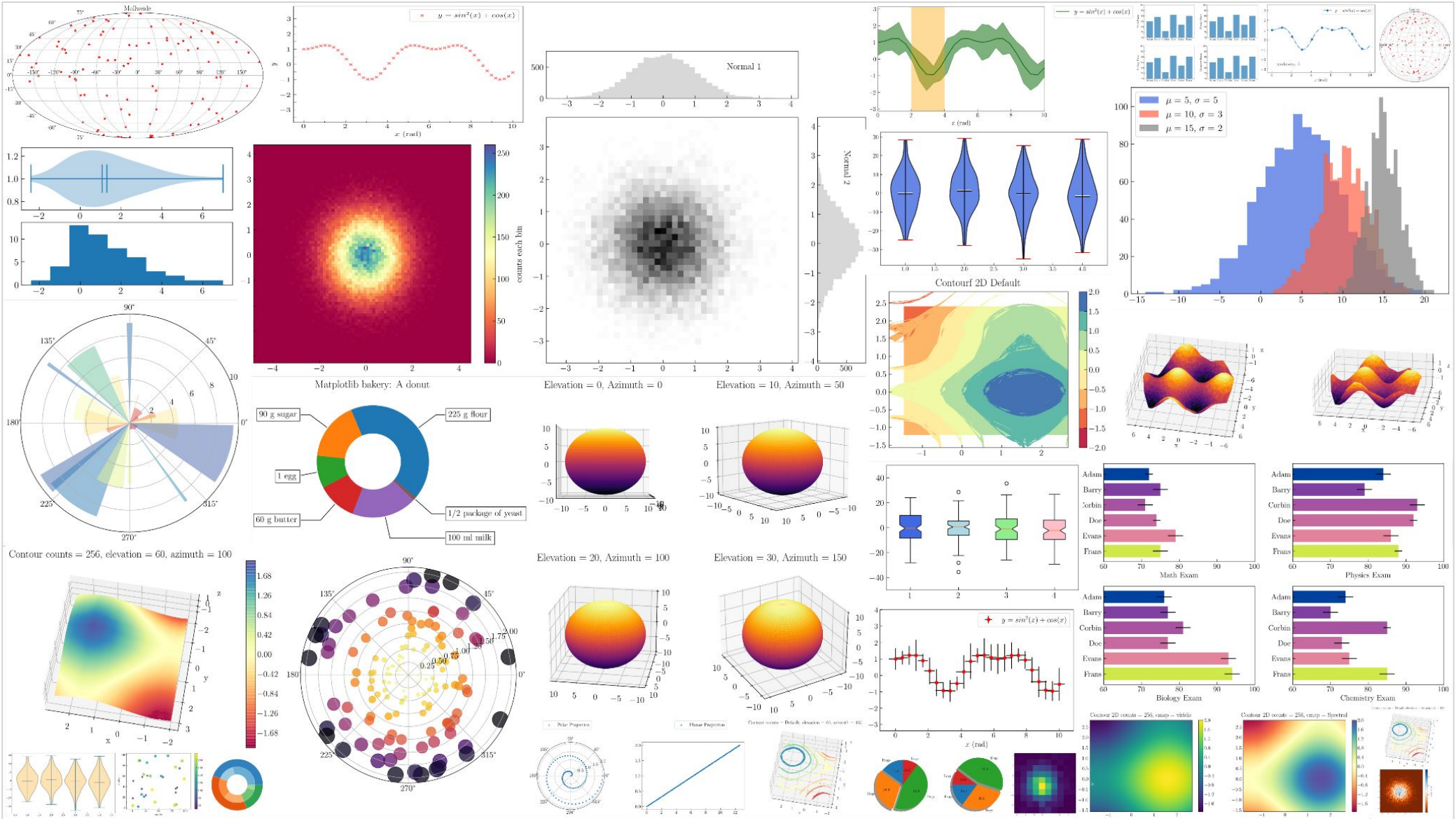
Nomenclaturas y conceptos clave: parámetros de estilo y ancho de línea (*style/line width*), tamaño de marcador (*marker size*), cuadrícula (*grid*), figura (*figure*), ejes (*axes*), funciones `plot()` y `subplot()`, trazado de múltiples curvas en un mismo gráfico, y nociones básicas de visualización en 3D.

matplotlib

Matplotlib

Python Data Visualization with Matplotlib

matplotlib



SciPy

SciPy (pronunciado "Sigh Pie") es una **librería** para la computación científica que amplía las capacidades de ***NumPy*** al proporcionar herramientas adicionales para el trabajo con matrices y estructuras de datos especializadas, como matrices dispersas y árboles k-dimensionales.

Ofrece **algoritmos fundamentales** para optimización, integración, interpolación, problemas de valores propios, ecuaciones algebraicas y diferenciales, estadísticas y muchas otras clases de problemas, siendo **ampliamente aplicable** en diversos dominios científicos y técnicos.

SciPy

Destaca por ser **performante**, ya que envuelve implementaciones altamente optimizadas escritas en lenguajes de bajo nivel como Fortran, C y C++, combinando la flexibilidad de Python con la velocidad del código compilado. Su **sintaxis de alto nivel** la hace **fácil de usar** y productiva para programadores de cualquier origen o nivel de experiencia.

Como proyecto de **código abierto**, *SciPy* se distribuye bajo una licencia BSD y es desarrollado y mantenido públicamente en **GitHub** por una comunidad vibrante, receptiva y diversa.

SciPy

SciPy es una colección de algoritmos matemáticos y funciones de conveniencia construidas en ***NumPy***. Añade un poder significativo a Python al proporcionar al usuario comandos y clases de alto nivel para manipular y visualizar datos. SciPy está organizado en subpaquetes que cubren diferentes dominios de computación científica. Estos se resumen en la siguiente tabla.

Tutorial de Python *SciPy*

Descripción del proyecto

Subpaquete	Descripción y guía del usuario
`cluster`	Algoritmos de agrupación
`constants`	Constantes físicas y matemáticas
`differentiate`	Herramientas de diferenciación de diferencias finitas
`fft`	Transformaciones de Fourier (scipy.fft)
`fftpack`	Rutinas de transformación rápida de Fourier (legado)
`integrate`	Integración (scipy.integrate)
`interpolate`	Interpolación (scipy.interpolate)
`io`	Archivo IO (scipy.io)
`linalg`	Álgebra lineal (scipy.linalg)
`ndimage`	Procesamiento de imágenes multidimensional (scipy.ndimage)
`odr`	Regresión de distancia ortogonal
`optimize`	Optimización (scipy.optimize)
`signal`	Procesamiento de señales (scipy.signal)
`sparse`	Matrices dispersas (scipy.sparse)
`spatial`	Estructuras de datos espaciales y algoritmos (scipy.spatial)
`special`	Funciones especiales (scipy.special)
`stats`	Estadísticas (scipy.stats)

Scikit-learn

Scikit-learn es un módulo de Python en **código abierto** para **aprendizaje automático (AA)** construido sobre ***NumPy***, ***Matplotlib*** y ***SciPy***; se distribuye bajo la licencia 3-Clause BSD.

El proyecto fue iniciado en **2007** por ***David Cournapeau*** como un proyecto de Google Summer of Code. Desde entonces, numerosos voluntarios han contribuido, y hoy es mantenido por un equipo dedicado de colaboradores.

Principales ventajas:

- Herramientas simples y eficientes para el análisis predictivo de datos
- Accesible para todos y reutilizable en varios contextos
- Código abierto, comercialmente utilizable

Scikit-learn

Herramientas - Guía de Usuário

Preprocesamiento: Extracción y normalización de características.

Aplicaciones: Transformar Datos de Entrada para su uso con Algoritmos de AA.
Algoritmos: Preprocesamiento, Extracción de Características y más...

Reducción de la dimensionalidad: Reducir el número de variables aleatorias a considerar.

Aplicaciones: Eliminación de Redundancias, Mayor eficiencia computacional, etc.
Algoritmos: PCA, Selección de Características, Factorización de Matrices no Negativas y más…

Selección de modelo: Comparar, validar y elegir parámetros y modelos.

Aplicaciones: Precisión mejorada a través del ajuste de parámetros.
Algoritmos: Búsqueda en Cuadrícula, Validación Cruzada, Métricas y más...

Scikit-learn

Herramientas - Guía de Usuário

Clasificación: Identificar a qué categoría pertenece un objeto.
Aplicaciones: Detección de Spam, Reconocimiento de Imágenes, etc.
Algoritmos: Aumento de Gradientes, Vecinos más Cercanos, Bosque Aleatorio, Regresión logística y más...

Regresión: Predicción de un atributo de valor continuo asociado a un objeto.
Aplicaciones: Respuesta a los Medicamentos, Precios de las Acciones, etc.
Algoritmos: Aumento de gradientes, Vecinos más cercanos, Bosque Aleatorio, Cresta y más...

Agrupación: Agrupación automática de objetos similares en conjuntos.
Aplicaciones: Segmentación de Clientes, Agrupación de Resultados de Experimentos, etc.
Algoritmos: k-Means, HDBSCAN, Agrupación Jerárquica y más...

scikit learn contrib

Clase 2: NumPy, Pandas, Matplotlib, SciPy y Scikit-learn