# Módulo de Programación en Python

Módulo TyHM

**Berard, Louise**
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
louise.berard28@gmail.com

**Ibañez, Celina**
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
mcelinaibanez@gmail.com

**Mobilia, Pilar**
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
pilarmobilia@gmail.com

**Torresi, Carla**
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
torresi.carla16@gmail.com

**Valli, Karima**
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
karimavallillalen@gmail.com

**Zanella, Bernardita**
Técnicas y Herramientas Modernas
Facultad de Ingeniería, Uncuyo
Mendoza, Argentina
bernizanella2014@gmail.com

Año 2024

## Abstract

El informe proporciona una visión general de Python y su utilidad en operaciones matemáticas y análisis de datos a través de bibliotecas clave como NumPy, Pandas y Matplotlib. Se explora la manipulación de listas y matrices en Python, destacando la importancia de NumPy para operaciones avanzadas. Se detallan ejemplos de creación de arrays, operaciones aritméticas y visualizaciones con Matplotlib. Además, se presenta un caso práctico de análisis de datos de calidad del vino utilizando Pandas y Matplotlib. En resumen, Python y sus bibliotecas ofrecen herramientas poderosas para la ciencia de datos, facilitando tareas complejas de manera eficiente.

***K*eywords** Python · NumPy · Pandas · Matplotlib

## 1 Introducción

Python se ha convertido en un lenguaje de programación ampliamente utilizado debido a su facilidad de interpretación, dinamismo y simplicidad. En el ámbito de las operaciones matemáticas y el análisis de datos, Python ofrece un conjunto de bibliotecas fundamentales como NumPy, Pandas y Matplotlib, que permiten realizar tareas avanzadas de manera eficiente. Este informe se centra en explorar el uso de estas bibliotecas, desde la manipulación de listas y matrices hasta la visualización de datos, destacando su importancia en la ciencia de datos. A través de ejemplos prácticos y explicaciones detalladas, se mostrará cómo Python y sus bibliotecas proporcionan herramientas poderosas para la manipulación, análisis y visualización de datos en diversos contextos.

## 2 ¿Qué es Python?

Python es un lenguaje de programación, el cual es muy utilizado debido a que es fácil de interpretar, dinámico y sencillo. A continuación, se resume la información sobre algunas de las bibliotecas más importantes de Python y su uso en operaciones matemáticas y análisis de datos.

### 2.1 Listas y Matrices en Python

Las listas en Python son estructuras de datos que permiten almacenar múltiples elementos. Pueden ser manipuladas para obtener información específica mediante el uso de índices. Es decir, contiene información distinta en cada lugar y cada posición de la lista me brinda lo que contiene.

Por ejemplo, si mi_lista = [1, 2, 3], entonces mi_lista[1] devuelve el número 2.

Además, Python permite crear matrices como listas de listas.

Por ejemplo, mi_matriz = [[1, 2, 3], [4, 5, 6]], lo que resulta en una matriz 2x3.

### 2.2 Bibliotecas de Python

Son paquetes con funciones y métodos que le permiten al usuario hacer operaciones con mayor facilidad.

#### 2.2.1 NumPy

Es una biblioteca fundamental para operaciones matemáticas avanzadas en Python. Permite realizar operaciones vectorizadas, trabajar con arrays (arreglos) n-dimensionales y matrices, y ejecutar operaciones matemáticas de manera eficiente.

• Creación de arrays: np.array(mi_lista) convierte una lista en un array de NumPy. Las operaciones aritméticas se pueden realizar directamente sobre los arrays.

Por ejemplo,

```
mi_lista = [1, 2, 3]
Mi_vector=np.ndarray(mi_lista)
Type(mi_vector)
Numpy.ndarray
3*mi_vector
Array[3,6,9] → cada elemento de mi_vector lo multiplica por 3.
3*mi_lista
[1,2,3,1,2,3,1,2,3] → crea una lista de 3 veces esa lista.
```

• Generación de arrays: np.zeros(10) crea un array de ceros y longitud 10, y np.ones(10) crea un array de unos. Otra forma de crear unarray es con el range, por ejemplo, A=np.range(0,10,1) va desde el 0 hasta el 10 con un paso de 1.

• Operaciones con arrays: Arrays pueden ser sumados, multiplicados, y manipulados mediante operaciones como np.sqrt(4) para obtener la raíz cuadrada. Además, un array de un array es una matriz.

```
suma = np.sum(mi_array)
raiz_cuadrada = np.sqrt(16)
Zeros[7]=4 → al elemento 7 del vector zeros, le pone un 4
```

#### 2.2.2 Pandas

Es una biblioteca utilizada para la manipulación y el análisis de datos tabulares. Trabaja con estructuras de datos denominadas DataFrames, que son similares a las tablas en bases de datos o a las hojas de cálculo, donde las columnas son las características y las filas son las observaciones.

• Lectura de datos: pd.read_csv(path) se usa para leer archivos CSV y convertirlos en DataFrames.

```
import pandas as pd
df = pd.read_csv('archivo.csv')
```

• Análisis y estadísticas: para leer un data frame se puede utilizar los siguientes códigos:

```
Df.tail() → da los últimos valores
Df.sample(3) → da 3 valores aleatorios
Df → da los primeros y los últimos, es como un head y tail juntos
Df.colums → da los features
Df.index→ el índice es lo que hace único a mi fila, esta entrada es única
```

• Manipulación de datos: Nuevas columnas pueden ser creadas, y los datos pueden ser filtrados y transformados. Por ejemplo, df[‘nueva_columna’] = df[‘columna’].apply(lambda x: . . . ) aplica una función a cada elemento de una columna.

### 2.2.3 Matplotlib

Es una biblioteca para la creación de gráficos y visualizaciones.

- Creación de gráficos: plt.plot(x, y) crea un gráfico de líneas de los datos en x y

y. plt.show() muestra el gráfico.

```
import matplotlib.pyplot as plt
Plt.plot(x,y)
plt.show()
```

- Personalización: Se pueden agregar etiquetas a los ejes (plt.xlabel y plt.ylabel), títulos (plt.title), y ajustar los límites de los ejes (plt.xlim y plt.ylim).

```
import matplotlib.pyplot as plt
Fig=plt.figure()
Axis.set_xlabel(´cant de h´)
Axis.set_ylabel(´consumo $´)
Axis.set_title(´consumo de motor´)
Axis.set_xlim(0,100)
Axis.set_ylim(0,2000)
Axis.plot(x,y)
```

- Figuras y ejes: fig = plt.figure() y ax = fig.add_axes([0, 0, 1, 1]) permiten crear gráficos personalizados con ejes específicos.

```
import matplotlib.pyplot as plt
Fig=plt.figure()
Axis=fig.add_axes([0,0,1,1])
Axis.set_xlabel(´cant de h´)
Axis.set_ylabel(´consumo $´)
Axis.set_title(´consumo de motor´)
Axis.set_xlim(0,100)
Axis.set_ylim(0,2000)
Axis.plot(x,y)
```

## 2.3 Ejemplo de Uso en Ciencia de Datos

Un ejemplo práctico de cómo se usan estas bibliotecas en conjunto es el análisis de un conjunto de datos de calidad del vino. A través de Pandas, se puede cargar y explorar el conjunto de datos, realizar transformaciones como la creación de nuevas columnas para clasificar la calidad del vino, y finalmente usar Matplotlib para visualizar los resultados.

Por ejemplo,

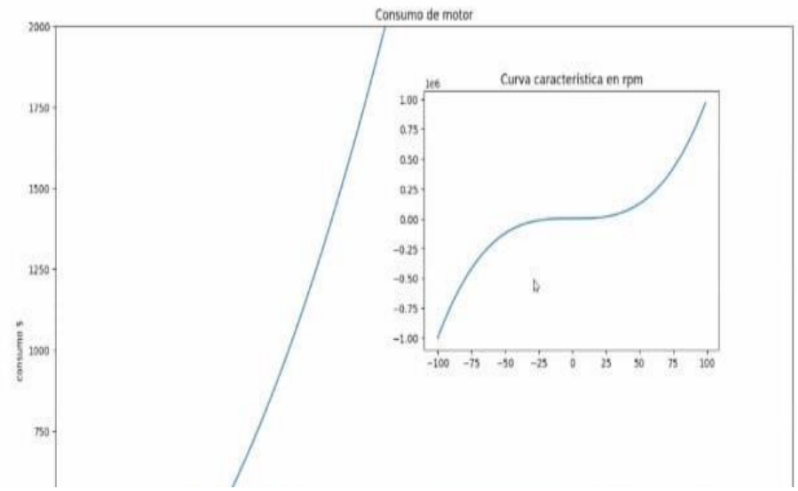


Figure 1: Figuras y ejes en Matplotlib

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

#### 2.3.1 Cargar datos

```
df = pd.read_csv('winequality-red.csv')
```

#### 2.3.2 Análisis preliminar

```
print(df.head())
print(df.describe())
```

#### 2.3.3 Transformación de datos

```
df['quality_label'] = df['quality'].apply(lambda x: 'bueno' si x >= 7 else 'malo')
```

#### 2.3.4 Visualización

```
plt.hist(df['quality_label'])
plt.title('Distribución de la Calidad del Vino')
plt.xlabel('Calidad')
plt.ylabel('Frecuencia')
plt.show()
```

## 3 Conclusión

Python, con sus bibliotecas como NumPy, Pandas y Matplotlib, proporciona un potente conjunto de herramientas para la manipulación, análisis y visualización de datos. Su uso es esencial en la ciencia de datos, permitiendo realizar tareas complejas de manera eficiente y con menos código.