Data Analytics with Python Quick Start Guide

A step-by-step tutorial covering pandas, numpy, and data visualization basics

By Ayoolumi Melehon
ayofemimelehon.com

1. Setting Up Your Environment

Installing Python

Download Python from python.org or use Anaconda (recommended for data science).

Installing Required Libraries

```
pip install pandas numpy matplotlib seaborn jupyter
```

Starting Jupyter Notebook

```
jupyter notebook
```

This opens a browser where you can write and run Python code interactively.

Importing Libraries

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

■ Convention: We use 'pd' for pandas and 'np' for numpy. You'll see this everywhere.

2. Pandas Fundamentals

Creating a DataFrame

```
# From a dictionary
data = {
'Name': ['Alice', 'Bob', 'Charlie'],
'Age': [25, 30, 35],
'City': ['London', 'Paris', 'Berlin']
}
df = pd.DataFrame(data)
print(df)
```

Reading Data from Files

```
# CSV file
df = pd.read_csv('data.csv')

# Excel file
df = pd.read_excel('data.xlsx')

# With options
df = pd.read_csv('data.csv', sep=',', header=0, encoding='utf-8')
```

Exploring Your Data

```
df.head() # First 5 rows
df.tail(10) # Last 10 rows
df.shape # (rows, columns)
df.info() # Column types and null counts
df.describe() # Statistics for numeric columns
df.columns # Column names
df.dtypes # Data types
```

3. Selecting & Filtering Data

Selecting Columns

```
# Single column (returns Series)
df['Name']

# Multiple columns (returns DataFrame)
df[['Name', 'Age']]

# Using dot notation (single column)
df.Name
```

Selecting Rows

```
# By index position (iloc)
df.iloc[0] # First row
df.iloc[0:5] # First 5 rows
df.iloc[-1] # Last row

# By label (loc)
df.loc[0] # Row with index 0
df.loc[0:5, 'Name'] # Rows 0-5, Name column
```

Filtering with Conditions

```
# Single condition
df[df['Age'] > 25]

# Multiple conditions (AND)
df[(df['Age'] > 25) & (df['City'] == 'London')]

# Multiple conditions (OR)
df[(df['City'] == 'London') | (df['City'] == 'Paris')]

# Using isin()
df[df['City'].isin(['London', 'Paris'])]

# String contains
df[df['Name'].str.contains('Ali')]
```

■ Always wrap conditions in parentheses and use & for AND, | for OR.

4. Data Manipulation

Adding New Columns

```
# Simple assignment
df['Country'] = 'UK'

# Based on calculation
df['Age_in_10_years'] = df['Age'] + 10

# Based on condition
df['Is_Adult'] = df['Age'] >= 18
```

Modifying Values

```
# Replace values
df['City'].replace('London', 'Greater London', inplace=True)

# Using loc for specific rows
df.loc[df['Age'] > 30, 'Category'] = 'Senior'
```

Handling Missing Data

```
# Check for missing values
df.isnull().sum()

# Drop rows with any missing values
df.dropna()

# Fill missing values
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['City'].fillna('Unknown', inplace=True)
```

5. Grouping & Aggregation

Basic Aggregations

```
df['Sales'].sum() # Total
df['Sales'].mean() # Average
df['Sales'].median() # Median
df['Sales'].min() # Minimum
df['Sales'].max() # Maximum
df['Sales'].count() # Count (non-null)
df['Sales'].std() # Standard deviation
```

Group By

```
# Single column grouping
df.groupby('City')['Sales'].sum()

# Multiple aggregations
df.groupby('City')['Sales'].agg(['sum', 'mean', 'count'])

# Multiple columns
df.groupby(['City', 'Year'])['Sales'].sum()

# Different aggregations per column
df.groupby('City').agg({
'Sales': 'sum',
'Customers': 'count',
'Profit': 'mean'
})
```

Pivot Tables

```
pd.pivot_table(
df,
values='Sales',
index='City',
columns='Year',
aggfunc='sum'
)
```

6. NumPy Essentials

Creating Arrays

```
arr = np.array([1, 2, 3, 4, 5])
zeros = np.zeros(10) # Array of zeros
ones = np.ones((3, 4)) # 3x4 matrix of ones
range_arr = np.arange(0, 10, 2) # [0, 2, 4, 6, 8]
linspace = np.linspace(0, 1, 5) # 5 evenly spaced from 0-1
```

Array Operations

```
arr * 2 # Multiply all by 2
arr + 10 # Add 10 to all
arr ** 2 # Square all
np.sqrt(arr) # Square root
np.log(arr) # Natural log

# Statistics
arr.mean(), arr.std(), arr.min(), arr.max()
```

7. Data Visualization

Basic Plots with Matplotlib

```
# Line plot
plt.plot(df['Date'], df['Sales'])
plt.xlabel('Date')
plt.ylabel('Sales')
plt.title('Sales Over Time')
plt.show()

# Bar chart
plt.bar(df['Category'], df['Sales'])
plt.show()

# Histogram
plt.hist(df['Age'], bins=10)
plt.show()

# Scatter plot
plt.scatter(df['Age'], df['Income'])
plt.show()
```

Better Plots with Seaborn

```
# Distribution plot
sns.histplot(df['Sales'], kde=True)

# Box plot
sns.boxplot(x='Category', y='Sales', data=df)

# Scatter with regression line
sns.regplot(x='Age', y='Income', data=df)

# Heatmap (correlation matrix)
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

# Count plot (bar chart for categories)
sns.countplot(x='City', data=df)
```

Customizing Plots

```
# Figure size
plt.figure(figsize=(10, 6))

# Style
sns.set_style('whitegrid')

# Colors
plt.plot(x, y, color='teal', linewidth=2)

# Save figure
plt.savefig('my_plot.png', dpi=300, bbox_inches='tight')
```

8. Complete Example: Sales Analysis

```
# Load data
df = pd.read_csv('sales_data.csv')

# Quick exploration
print(df.head())
print(df.info())
```

```
# Clean data
df.dropna(inplace=True)
df['Date'] = pd.to_datetime(df['Date'])

# Add month column
df['Month'] = df['Date'].dt.month

# Monthly sales summary
monthly_sales = df.groupby('Month')['Revenue'].agg(['sum', 'mean', 'count'])
print(monthly_sales)

# Top 5 products
top_products = df.groupby('Product')['Revenue'].sum().nlargest(5)
print(top_products)

# Visualize
plt.figure(figsize=(10, 5))
sns.barplot(x=top_products.index, y=top_products.values)
plt.title('Top 5 Products by Revenue')
plt.ylabel('Revenue (£)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('top_products.png')
plt.show()
```

■ Practice makes perfect! Try these techniques on your own datasets.