Python Data Analysis Reference Guide

by Manu Gupta

FH Joanneum | October 2024

Contents

1 Python Fundamentals for Data Analysis

1.1 Jupyter Notebook Essentials

Keyboard Shortcuts

- **Shift + Enter:** Execute cell
- **Esc + m:** Change to Markdown cell
- **Esc + y:** Change to Code cell
- **Ctrl + Enter:** Run cell in place

1.2 Markdown Basics

```
# Heading 1
## Heading 2
**Bold Text**
*Italic Text*
[Link Text](URL)
```

1.3 Data Types

- **Integer:** -256, 15
- **Float:** -253.23, 1.253e-10
- **String:** "Hello", 'Goodbye'
- **Boolean:** True, False
- **List:** [value, ...]
- **Dictionary:** {key: value}

1.4 Basic Python Operations

1.4.1 Arithmetic Operators

Arithmetic Operations

- x + y (add)
- x - y (subtract)
- x * y (multiply)
- x / y (divide)
- x % y (modulus)
- x ** y (power)

Assignment shortcuts: x op= y
Example: x += 1 increments x

1.4.2 Comparison Operators

Comparison Operations

- x < y (less than)
- x <= y (less or equal)
- x > y (greater than)
- x >= y (greater or equal)
- x == y (equal)
- x ! = y (not equal)

1.4.3 Boolean Operators

Boolean Operations

- not x (logical NOT)
- x and y (logical AND)
- x or y (logical OR)

1.4.4 If-Else Statements

```
# Basic if statement
if expression:
    statements
elif expression:
    statements
else:
    statements

# Example
score = 85
if score >= 90:
    grade = 'A'
elif score >= 80:
    grade = 'B'
else:
    grade = 'C'
```

1.4.5 List Operations

Common List Operations

```
# List creation and modification
lst = [1, 2, 3, 4, 5]

del lst[i]          # Deletes ith item from lst
lst.append(e)       # Appends e to lst
lst.insert(i, e)    # Inserts e before ith item in lst
lst.sort()          # Sorts lst
lst.reverse()       # Reverses lst
lst.pop()           # Removes and returns last element
lst.index(x)        # Returns index of first x
lst.count(x)        # Counts occurrences of x
```

1.5 Control Structures

1.5.1 List Comprehension

```
squares = [x**2 for x in range(10)]
evens = [x for x in range(10) if x % 2 == 0]
```

1.5.2 Loops

```
# For Loop
for i in range(start, end, step):
    print(i)

# While Loop
```

```
count = 0
while count < 5:
    print(count)
    count += 1
```

1.6 Functions and Classes

```
# Function Definition
def calculate_mean(numbers):
    return sum(numbers) / len(numbers)

# Class Definition
class DataPoint:
    def __init__(self, value):
        self.value = value

    def transform(self):
        return self.value ** 2
```

2 Data Analysis Tools

2.1 NumPy Essentials

```
import numpy as np

# Array Creation
arr = np.array([1, 2, 3])
zeros = np.zeros((2, 3))
ones = np.ones((3, 2))
identity = np.eye(3)

# Array Operations
mean = np.mean(arr)
std = np.std(arr)
sum = np.sum(arr)
```

2.2 Pandas Fundamentals

```
import pandas as pd

# Reading Data
df = pd.read_csv("file.csv")

# Basic Operations
df.head()  # First 5 rows
df.describe()  # Statistical summary
df.info()  # DataFrame information

# Data Cleaning
df.dropna(inplace=True)  # Remove rows with missing values
df.fillna(0, inplace=True)  # Fill missing values
#inplace=True, in order to modifying the already existing dataframe.

# Column Operations
df['new_column'] = df['col1'] + df['col2']
df.rename(columns={'old': 'new'}, inplace=True)
df.drop(['Column name'], axis=1, inplace=True) # Remove a column
pd.to_numeric(df['Temperature']) # Convert the column data type (e.g., string) to
    numeric

# Date Operations
df['date'] = pd.to_datetime(df['date']) # Convert the column to date time format
```

2.3 Data Visualization with Matplotlib

```
import matplotlib.pyplot as plt

# Line Plot
plt.plot(x, y, marker='o')
plt.xlabel('Time')
plt.ylabel('Value')
plt.title('Time Series Data')

# Bar Plot
plt.bar(categories, values)
plt.show()

# Multiple Plots
fig, (ax1, ax2) = plt.subplots(1, 2)
ax1.plot(x1, y1)
ax2.scatter(x2, y2)
```

2.4 Random Data Generation

```
import random

# Random Numbers
random.random()  # [0, 1)
random.uniform(0, 10)  # [0, 10]
random.gauss(10, 2)  # Gaussian distribution
random.randint(1, 100)  # Integer [1, 100]

# Random Selections
random.choice(['apple', 'banana', 'cherry'])
random.sample(range(100), 10)

# Set Seed
random.seed(42)
```

3 Data Analysis Best Practices

Python Data Analysis Tips

- Use vectorized operations (NumPy arrays) instead of loops when possible
- Create functions for repetitive analysis tasks
- Keep original data unchanged, work with copies
- Use appropriate data types for statistical calculations
- Handle missing values appropriately for your analysis
- Document your analysis pipeline
- Use version control for your analysis scripts

Statistical Analysis Tips

- Always check data distribution before applying statistical tests
- Use appropriate visualization for different types of data
- Check for outliers and their impact on analysis
- Validate assumptions of statistical models
- Document data cleaning and transformation steps
- Use appropriate measures of central tendency
- Consider the sample size when drawing conclusions

4 Useful Resources

- Python Documentation: `https://docs.python.org`
- Pandas Documentation: `https://pandas.pydata.org/docs`
- NumPy Documentation: `https://numpy.org/doc`
- Matplotlib Documentation: `https://matplotlib.org`