25 PYTHON DATA ANALYSIS EXERCISES

Complete Answer Guide

Five progressive learning modules. Work independently before reading the provided solutions. Data are invented; not actual student or environmental measurements.

MODULE 1: PYTHON FUNDAMENTALS

Exercise 1: Calculate the Average Examination Score (Beginner)

Learning objective

Calculate a mean from a Python list using sum() and len().

Problem and sample data

Five fictional test scores are provided. Find their arithmetic average and describe what it represents. Data: scores = [62, 75, 81, 92, 70]

Verified solution

```
scores = [62,75,81,92,70]
average = sum(scores)/len(scores)
print(f"Mean: {average:.1f}")
```

Verified output

```
Mean: 76.0
```

Why it works

Sum the five values to get 380, then divide by five. A mean can be pulled upward by a very high score, so inspect the distribution too.

Common mistake

Dividing by a fixed number when the list length changes.

Extension task

Replace the highest score with 100 and recalculate the average.

Exercise 2: Find Maximum and Minimum Daily Temperatures (Beginner)

Learning objective

Use min() and max() to summarize extremes.

Problem and sample data

Identify the coolest and warmest fictional day. Data: temps = [21,24,19,27,23] °C

Verified solution

```
temps = [21,24,19,27,23]
print('Coolest:',min(temps),'°C')
print('Warmest:',max(temps),'°C')
```

Verified output

```
Coolest: 19 °C
Warmest: 27 °C
```

Why it works

The extremes provide a quick range check, but the five days cannot establish a climate trend.

Common mistake

Confusing the positions of values with their temperatures.

Extension task

Calculate the range (maximum minus minimum).

Exercise 3: Count Scores Reaching a Threshold (Beginner)

Learning objective

Use a loop and an if condition to count qualifying observations.

Problem and sample data

Count fictional scores of at least 75, including scores exactly equal to 75. Data: scores = [48,65,75,90,59,75]

Verified solution

```
scores = [48,65,75,90,59,75]
count=0
for score in scores:
    if score&gt;=75:
        count+=1
print('At least 75:',count)
```

Verified output

```
At least 75: 3
```

Why it works

Two scores equal 75 and one is 90, so three pass this illustrative threshold.

Common mistake

Using > rather than >= and excluding a score of exactly 75.

Extension task

Count values strictly greater than 75 and explain the change.

Exercise 4: Calculate Electricity Use Percentage Change (Beginner)

Learning objective

Compute percentage change using an initial value and a new value.

Problem and sample data

A fictional household used 240 kWh in one month and 198 kWh the next. Find percentage change. Data: old = 240 kWh; new = 198 kWh

Verified solution

```
old,new=240,198
change=(new-old)/old*100
print(f'Change: {change:.1f}%')
```

Verified output

```
Change: -17.5%
```

Why it works

Use (new-old)/old × 100; the negative result represents a 17.5% decrease. Never divide by zero.

Common mistake

Using the new value as the denominator rather than the original.

Extension task

Guard against old=0 and describe why percentage change is then undefined.

Exercise 5: Write a Reusable Mean Function (Beginner)

Learning objective

Define a function with a parameter and return value; handle empty inputs.

Problem and sample data

Write a function to average any non-empty numeric list and test it with three values. Data: values=[3,6,9]

Verified solution

```
def safe_mean(values):
    if not values:
        raise ValueError('Need at least one value')
    return sum(values)/len(values)
print(f'Mean: {safe_mean([3,6,9]):.1f}')
```

Verified output

```
Mean: 6.0
```

Why it works

A function prevents repeated arithmetic code. It raises a clear error when asked to average zero values.

Common mistake

Silently returning zero for an empty dataset, which is not its mean.

Extension task

Write a second function returning both the mean and sample size.

MODULE 2: NUMPY

Exercise 6: Create and Inspect a NumPy Array (Developing)

Learning objective

Learn array construction, shape, size and dimension count.

Problem and sample data

Create an array of five fictional rainfall measurements. Report its structure. Data: rain=[12,5,0,18,9] mm

Verified solution

```
import numpy as np
rain=np.array([12,5,0,18,9])
print('shape:',rain.shape,'size:',rain.size,'ndim:',rain.ndim)
```

Verified output

```
shape: (5,) size: 5 ndim: 1
```

Why it works

A one-dimensional array of length five has a shape tuple (5,), a total size of five and one dimension.

Common mistake

Calling shape a number of dimensions rather than a tuple of sizes.

Extension task

Create a two-row array and compare ndim and shape.

Exercise 7: Convert Celsius Temperatures to Fahrenheit (Developing)

Learning objective

Apply scalar arithmetic to all elements of a NumPy array.

Problem and sample data

Convert three fictional Celsius readings to Fahrenheit using vectorization. Data: Celsius: [0,20,37]

Verified solution

```
import numpy as np
c=np.array([0,20,37])
f=c*9/5+32
print('Fahrenheit:',f.tolist())
```

Verified output

```
Fahrenheit: [32.0, 68.0, 98.6]
```

Why it works

The formula F = 9C/5 + 32 applies elementwise without a Python loop.

Common mistake

Applying +32 before scaling by 9/5.

Extension task

Convert -10°C and test whether the output is 14°F.

Exercise 8: Find NumPy Mean, Median and Standard Deviation (Developing)

Learning objective

Compare central tendency and population/sample spread.

Problem and sample data

Compute mean, median, population SD and sample SD for four numbers. Data: x=[2,4,6,8]

Verified solution

```
import numpy as np
x=np.array([2,4,6,8])
print('Mean:',np.mean(x))
print('Median:',np.median(x))
print('Population SD:',round(np.std(x,ddof=0),4))
print('Sample SD:',round(np.std(x,ddof=1),4))
```

Verified output

```
Mean: 5.0
Median: 5.0
Population SD: 2.2361
Sample SD: 2.582
```

Why it works

Population SD divides squared deviations by n; sample SD uses n–1. State the convention used.

Common mistake

Treating NumPy's default ddof=0 as sample SD.

Extension task

Change the data to [2,4,6,20] and compare the mean with the median.

Exercise 9: Select Readings Above a Fictional Limit (Developing)

Learning objective

Use Boolean indexing to find selected values.

Problem and sample data

From a fictional environmental sensor, select readings strictly above 40 units. This is not a safety threshold. Data: readings=[21,35,44,18,55]

Verified solution

```
import numpy as np
readings=np.array([21,35,44,18,55])
selected=readings[readings&gt;40]
print('Above 40:',selected.tolist())
```

Verified output

```
Above 40: [44, 55]
```

Why it works

The comparison produces a Boolean mask, which selects only entries at matching positions.

Common mistake

Calling a fictional exercise threshold an official public-health standard.

Extension task

Count qualifying readings and express the fraction of all five.

Exercise 10: Reshape Twelve Observations into Four Weeks (Developing)

Learning objective

Reshape an array while preserving the element count.

Problem and sample data

Twelve observations were collected three days per week for four weeks. Form a 4×3 array and calculate weekly totals. Data: values=1 to 12

Verified solution

```
import numpy as np
values=np.arange(1,13)
weeks=values.reshape(4,3)
print('Shape:',weeks.shape)
print('Weekly totals:',weeks.sum(axis=1).tolist())
```

Verified output

```
Shape: (4, 3)
Weekly totals: [6, 15, 24, 33]
```

Why it works

A 4×3 array contains 12 entries. axis=1 sums across each week’s three columns.

Common mistake

Attempting to reshape 12 values into a shape with a different total size.

Extension task

Compute daily totals across four weeks using axis=0.

MODULE 3: PANDAS

Exercise 11: Build a Student Pandas DataFrame (Developing)

Learning objective

Create a labeled table with distinct rows and columns.

Problem and sample data

Represent four invented learners, subjects and scores in a DataFrame. Data: Asha Math 78; Leo Science 85; Mina Math 91; Noah Science 74

Verified solution

```
import pandas as pd
df=pd.DataFrame({'Student':['Asha','Leo','Mina','Noah'],
 'Subject':['Math','Science','Math','Science'],
 'Score':[78,85,91,74]})
print('Shape:',df.shape)
print('Columns:',df.columns.tolist())
```

Verified output

```
Shape: (4, 3)
Columns: ['Student', 'Subject', 'Score']
```

Why it works

Each list becomes a column; matching list lengths produce four complete rows. The default row index is not a student ID.

Common mistake

Assuming the DataFrame index is always a meaningful source identifier.

Extension task

Add an Age column with four fictional ages.

Exercise 12: Filter Students Who Scored at Least 80 (Developing)

Learning objective

Apply a Boolean mask to a Pandas DataFrame.

Problem and sample data

Select student names with scores of at least 80 from the same four-row table. Data: Asha 78; Leo 85; Mina 91; Noah 74

Verified solution

```
import pandas as pd
df=pd.DataFrame({'Student':['Asha','Leo','Mina','Noah'],
 'Subject':['Math','Science','Math','Science'],
 'Score':[78,85,91,74]})
selected=df.loc[df.Score&gt;=80,'Student'].tolist()
print('Selected:',selected)
```

Verified output

```
Selected: ['Leo', 'Mina']
```

Why it works

Both Leo and Mina meet the numerical rule; Asha and Noah do not.

Common mistake

Using the wrong comparison and filtering out exactly 80.

Extension task

Change the threshold to 75 and check the new selection.

Exercise 13: Sort Examination Results from Highest to Lowest (Developing)

Learning objective

Use sort_values to order records without confusing order and ranking.

Problem and sample data

Sort the fictional four-row table descending by Score. Data: Scores: Asha 78; Leo 85; Mina 91; Noah 74

Verified solution

```
import pandas as pd
df=pd.DataFrame({'Student':['Asha','Leo','Mina','Noah'],
 'Score':[78,85,91,74]})
ordered=df.sort_values('Score',ascending=False)
print('Order:',ordered.Student.tolist())
```

Verified output

```
Order: ['Mina', 'Leo', 'Asha', 'Noah']
```

Why it works

Sorting reorders rows; numerical ranks require additional decisions about tied scores.

Common mistake

Calling the original row index a rank after sorting.

Extension task

Add a tied score and explore rank(method="min").

Exercise 14: Compute Average Scores by Subject (Developing)

Learning objective

Group observations by category and calculate group means.

Problem and sample data

Find the Math and Science means for the shared fictional student table. Data: Math 78,91; Science 85,74

Verified solution

```
import pandas as pd
df=pd.DataFrame({'Subject':['Math','Science','Math','Science'],
 'Score':[78,85,91,74]})
means=df.groupby('Subject').Score.mean()
print({k:float(v) for k,v in means.items()})
```

Verified output

```
{'Math': 84.5, 'Science': 79.5}
```

Why it works

Groupby separates rows by subject and averages each pair. Two observations per group are not a basis for population conclusions.

Common mistake

Calculating the overall mean when a subject-level average is requested.

Extension task

Also report count and minimum by subject.

Exercise 15: Calculate Percentage Scores in a New Column (Developing)

Learning objective

Use vectorized Pandas arithmetic with row-specific denominators.

Problem and sample data

Three fictional tasks have different maximum marks. Calculate earned percentages. Data: earned=[76,60,45]; maximum=[100,80,50]

Verified solution

```
import pandas as pd
df=pd.DataFrame({'Earned':[76,60,45],
 'Maximum':[100,80,50]})
assert df.Maximum.gt(0).all()
df['Percentage']=100*df.Earned/df.Maximum
print('Percentages:',df.Percentage.tolist())
```

Verified output

```
Percentages: [76.0, 75.0, 90.0]
```

Why it works

Each row has its own denominator, so dividing all marks by 100 would misrepresent the other tasks.

Common mistake

Ignoring zero or different maximum marks.

Extension task

Compute average percentage across the three tasks and explain its meaning.

MODULE 4: CSV AND CLEANING

Exercise 16: Read a Reproducible CSV File (Developing)

Learning objective

Write then read a small CSV and inspect its structure.

Problem and sample data

Create the supplied rainfall CSV, then load it using read_csv. Data: TownA 18; TownB 24; TownC 11; TownD 30 (mm)

Verified solution

```
from pathlib import Path
import pandas as pd
file=Path('town-rainfall.csv')
pd.DataFrame({'Town':['TownA','TownB','TownC','TownD'],
 'RainMM':[18,24,11,30]}).to_csv(file,index=False)
loaded=pd.read_csv(file)
print('Shape:',loaded.shape)
print('Total mm:',loaded.RainMM.sum())
```

Verified output

```
Shape: (4, 2)
Total mm: 83
```

Why it works

Writing the file first makes the exercise reproducible without relying on an external download.

Common mistake

Assuming Python opens files from the browser download directory.

Extension task

Change the filename and check Path.cwd() before reading.

Exercise 17: Count Missing Values in Two Fields (Developing)

Learning objective

Use isna and sums to inspect completeness.

Problem and sample data

Count missing Score and Attendance entries in a fictional table. Data: Score=[80,None,72]; Attendance=[94,88,None]

Verified solution

```
import pandas as pd
df=pd.DataFrame({'Score':[80,None,72],
 'Attendance':[94,88,None]})
print(df.isna().sum().to_dict())
```

Verified output

```
{'Score': 1, 'Attendance': 1}
```

Why it works

One missing value occurs in each numeric column; a missing value is not automatically zero.

Common mistake

Treating a missing mark as a score of zero without justification.

Extension task

Describe one plausible reason why missingness may not be random.

Exercise 18: Normalize Inconsistent Subject Labels (Developing)

Learning objective

Clean spaces and map known abbreviations explicitly.

Problem and sample data

Standardize five category labels into Mathematics or Chemistry. Data: [" Mathematics ","math","MATH","Chemistry"," chemistry "]

Verified solution

```
import pandas as pd
labels=pd.Series([' Mathematics ','math','MATH',
                  'Chemistry',' chemistry '])
mapping={'mathematics':'Mathematics','math':'Mathematics',
         'chemistry':'Chemistry'}
clean=labels.str.strip().str.lower().map(mapping)
print(clean.value_counts().to_dict())
```

Verified output

```
{'Mathematics': 3, 'Chemistry': 2}
```

Why it works

Strip spaces, lowercase safely for matching, then map recognized labels. Unmatched values require review.

Common mistake

Assuming str.title converts MATH to Mathematics automatically.

Extension task

Add a previously unseen label and detect unmapped values.

Exercise 19: Remove Confirmed Exact Duplicate Rows (Developing)

Learning objective

Use duplicated() and drop_duplicates() after validating repeated records.

Problem and sample data

A small table contains the same A,8 record twice. Identify and remove only exact repetition. Data: A 8; B 6; A 8; C 7

Verified solution

```
import pandas as pd
df=pd.DataFrame({'ID':['A','B','A','C'],
 'Hours':[8,6,8,7]})
print('Exact duplicates:',int(df.duplicated().sum()))
clean=df.drop_duplicates()
print('Rows after cleaning:',len(clean))
```

Verified output

```
Exact duplicates: 1
Rows after cleaning: 3
```

Why it works

A repeated full row is a potential duplicate. Repeated IDs with different measurements may be legitimate separate records.

Common mistake

Removing every row with a repeated person ID.

Extension task

Add A with 9 hours and justify whether it is a duplicate.

Exercise 20: Export a Cleaned CSV and Read It Back (Developing)

Learning objective

Use to_csv(index=False) and verify the exported rows.

Problem and sample data

Write a cleaned three-record table and confirm its shape after rereading. Data: A 8; B 6; C 7

Verified solution

```
import pandas as pd
clean=pd.DataFrame({'ID':['A','B','C'],
 'Hours':[8,6,7]})
clean.to_csv('cleaned.csv',index=False,encoding='utf-8')
check=pd.read_csv('cleaned.csv')
print('Verified shape:',check.shape)
print('Total hours:',check.Hours.sum())
```

Verified output

```
Verified shape: (3, 2)
Total hours: 21
```

Why it works

Reimporting checks the file exists and that an unwanted extra index column was not written.

Common mistake

Overwriting the raw source data rather than producing a new file.

Extension task

Compare the reloaded DataFrame with clean using equals().

MODULE 5: CHARTS AND CAPSTONE

Exercise 21: Create a Labeled Bar Chart (Intermediate Beginner)

Learning objective

Use Figure and Axes to compare category magnitudes.

Problem and sample data

Plot three fictional subject averages as vertical bars and save the image. Data: English 78; Mathematics 73; Science 81

Verified solution

```
import matplotlib.pyplot as plt
names=['English','Mathematics','Science']
means=[78,73,81]
fig,ax=plt.subplots(figsize=(7,4))
ax.bar(names,means,color='#357b8b')
ax.set(title='Fictional subject means',ylabel='Mean / 100',ylim=(0,100))
fig.tight_layout();fig.savefig('exercise21-bar.png',dpi=150)
plt.close(fig)
print('Saved exercise21-bar.png')
```

Verified output

```
Saved exercise21-bar.png
```

Why it works

A zero-based vertical scale prevents exaggeration; include the score unit and note these are invented averages.

Common mistake

Omitting axis labels or truncating bar axes without justification.

Extension task

Add numeric labels to each bar.

Exercise 22: Plot Chronological Rainfall Changes (Intermediate Beginner)

Learning objective

Create a line plot with observations in time order.

Problem and sample data

Plot rainfall across five fictional months and save the figure. Data: Jan–May [42,50,31,62,55] mm

Verified solution

```
import matplotlib.pyplot as plt
months=['Jan','Feb','Mar','Apr','May']
rain=[42,50,31,62,55]
fig,ax=plt.subplots(figsize=(7,4))
ax.plot(months,rain,marker='o',color='#28774f')
ax.set(title='Fictional monthly rainfall',
       xlabel='Month',ylabel='Rainfall (mm)',ylim=(0,75))
fig.tight_layout();fig.savefig('exercise22-line.png',dpi=150)
plt.close(fig)
print('Saved exercise22-line.png')
```

Verified output

```
Saved exercise22-line.png
```

Why it works

A line helps reveal changes across ordered months, including a drop from February to March.

Common mistake

Connecting categories in arbitrary order and describing it as a time trend.

Extension task

Add a horizontal reference line at 50 mm.

Exercise 23: Plot Study Hours Against Examination Scores (Intermediate Beginner)

Learning objective

Use a scatter plot for two numeric variables and interpret cautiously.

Problem and sample data

Plot paired fictional observations without claiming causality. Data: Hours [1,2,3,4,5]; Scores [52,61,66,78,88]

Verified solution

```
import matplotlib.pyplot as plt
hours=[1,2,3,4,5]
scores=[52,61,66,78,88]
fig,ax=plt.subplots(figsize=(7,4))
ax.scatter(hours,scores,s=75,color='#385f9b')
ax.set(title='Fictional hours and scores',
       xlabel='Study hours / week',ylabel='Score / 100',
       xlim=(0,6),ylim=(40,100))
fig.tight_layout();fig.savefig('exercise23-scatter.png',dpi=150)
plt.close(fig)
print('Saved exercise23-scatter.png')
```

Verified output

```
Saved exercise23-scatter.png
```

Why it works

The constructed points rise together, but other variables could influence genuine examination results.

Common mistake

Treating visual association as proof of a causal study effect.

Extension task

Add a sixth observation that breaks the simple trend.

Exercise 24: Summarize Fictional Sales by Region (Intermediate Beginner)

Learning objective

Calculate revenue and group total and mean sale using groupby().agg().

Problem and sample data

Summarize a five-row synthetic shop dataset by North and South. Data: North: Book 3×15, Pen 5×2, Book 1×15; South: Book 2×15, Pen 4×2

Verified solution

```
import pandas as pd
sales=pd.DataFrame({'Region':['North','North','South','South','North'],
 'Item':['Book','Pen','Book','Pen','Book'],
 'Units':[3,5,2,4,1],
 'UnitPrice':[15,2,15,2,15]})
sales['Revenue']=sales.Units*sales.UnitPrice
summary=sales.groupby('Region').agg(
 TotalRevenue=('Revenue','sum'),
 Orders=('Revenue','size')).sort_index()
print(summary.to_dict('index'))
```

Verified output

```
{'North': {'TotalRevenue': 70, 'Orders': 3}, 'South': {'TotalRevenue': 38, 'Orders': 2}}
```

Why it works

Revenue is units multiplied by unit price per row; grouping then sums rows. Orders here means source rows, not necessarily customer orders.

Common mistake

Averaging unit prices rather than calculating actual sales revenue.

Extension task

Calculate average revenue per record and total units by region.

Exercise 25: Student Performance Data Analysis Challenge (Intermediate Beginner)

Learning objective

Combine data import, quality checks, grouping, charting and responsible interpretation.

Problem and sample data

Complete a full mini-project: create or import the fictional CSV, inspect missing values and duplicates, retain valid scored rows, calculate overall and subject means, identify scores of 80+, generate a chart and export a verified clean CSV. Data: 12 distinct scored fictional student records (P01–P12) across Math, Science and English; an extra exact repeat of P03 and a P13 row with missing score. Attendance for P06 is missing.

Verified solution

```
from pathlib import Path
import pandas as pd
import matplotlib.pyplot as plt
rows=[
('P01','Math',2,86,55),('P02','Math',3,90,63),
('P03','Math',4,94,72),('P04','Math',5,97,81),
('P05','Science',2,85,59),('P06','Science',3,None,68),
('P07','Science',5,95,80),('P08','Science',6,98,88),
('P09','English',1,84,61),('P10','English',3,89,70),
('P11','English',4,93,77),('P12','English',7,99,92)]
cols=['StudentID','Subject','StudyHours','AttendancePct','Score']
raw=pd.DataFrame(rows+[rows[2],('P13','English',2,87,None)],columns=cols)
raw.to_csv('exercise25-student-performance-raw.csv',index=False)
print('Raw shape:',raw.shape)
print('Exact duplicates:',int(raw.duplicated().sum()))
clean=raw.drop_duplicates().copy()
for col in ['StudyHours','AttendancePct','Score']:
    clean[col]=pd.to_numeric(clean[col],errors='coerce')
clean=clean.dropna(subset=['Score','StudyHours']).copy()
clean['AttendancePct']=clean.AttendancePct.fillna(clean.AttendancePct.median())
clean.to_csv('exercise25-student-performance-clean.csv',index=False)
print('Clean shape:',clean.shape)
print('Mean score:',round(clean.Score.mean(),4))
print('Group means:',clean.groupby('Subject').Score.mean().round(2).to_dict())
print('At least 80:',clean.loc[clean.Score.ge(80),'StudentID'].tolist())
means=clean.groupby('Subject').Score.mean()
fig,ax=plt.subplots(figsize=(7,4))
ax.bar(means.index,means.values,color='#307c87')
ax.set(title='Fictional mean examination scores',ylabel='Mean / 100',ylim=(0,100))
fig.tight_layout();fig.savefig('exercise25-capstone.png',dpi=150)
plt.close(fig)
print('Saved exercise25-capstone.png')
```

Verified output

```
Raw shape: (14, 5)
Exact duplicates: 1
Clean shape: (12, 5)
Mean score: 72.1667
Group means: {'English': 75.0, 'Math': 67.75, 'Science': 73.75}
At least 80: ['P04', 'P07', 'P08', 'P12']
Saved exercise25-capstone.png
```

Why it works

The verified project preserves the raw file, excludes the exact repeat and missing score, imputes one attendance gap with the median as a transparent teaching decision, calculates statistics, plots group means and exports the cleaned data. No correlation proves causation, and invented results cannot describe real learners.

Common mistake

Running group averages before cleaning the duplicate; silently changing missing scores to zero.

Extension task

Restart a new kernel, rerun the project and write a 100-word conclusion acknowledging its synthetic sample.