

Solutions

5.1 Reading and Writing Files

Reference example

```
from io import StringIO
with StringIO("red\nblue\n") as source:
    lines = [line.strip() for line in source]
print(lines)
```

5.1.1 - Explain

What can go wrong with opening an existing file in write mode when you intended to add records?

Write mode can truncate existing content. Choose the mode deliberately, preserve required data, and test changes on a safe example.

5.1.2 - Write code

Using StringIO, read the numbers on separate lines and print their total.

Parse each nonempty line as an integer.

```
from io import StringIO
with StringIO("4\n7\n") as source:
    total = sum(int(line) for line in source if line.strip())
print(total)
```

Expected output

11

Solutions

5.2 Structured Formats

Reference example

```
import json
payload = '{"name": "Ada", "scores": [4, 7]}'
data = json.loads(payload)
print(data["name"], sum(data["scores"]))
```

5.2.1 - Trace

Determine the exact output and explain the state changes.

```
import csv
from io import StringIO
rows = list(csv.DictReader(StringIO("name,score\nAda,7\n")))
print(rows[0]["score"], type(rows[0]["score"]).__name__)
```

Output

7 str

CSV reader fields are strings until the program performs appropriate conversion.

5.2.2 - Write code

Parse the JSON list [2, 5, 8] and print its largest value.

Load the structure, then use the appropriate list operation.

```
import json
values = json.loads("[2, 5, 8]")
print(max(values))
```

Expected output

8

Solutions

5.3 Consuming APIs

Reference example

```
response = {"status": 200, "body": {"items": ["A", "B"]}}
if response["status"] == 200:
    items = response["body"]["items"]
    print(len(items))
```

5.3.1 - Explain

Why should a client use a timeout and check the response status before processing an API body?

A request may fail or stall, and an error body may not match the success format. Handle transport and application errors explicitly.

5.3.2 - Write code

Process a local response whose body contains values [3, 6], and print their sum only when status is 200.

Check the status before using the success schema.

```
response = {"status": 200, "body": {"values": [3, 6]}}
if response["status"] == 200:
    print(sum(response["body"]["values"]))
```

Expected output

9

Solutions

5.4 Error Handling and Automation

Reference example

```
values = ["3", "bad", "5"]
total = 0
for value in values:
    try:
        total += int(value)
    except ValueError:
        print("skipped", value)
print(total)
```

5.4.1 - Explain

Why is an unrestricted except block risky in an automation script?

It can hide unrelated programming mistakes and make failed work look successful. Catch expected exceptions and report or propagate other failures.

5.4.2 - Write code

Convert ["2", "x", "4"] into integers, skipping invalid entries, and print the result.

Handle ValueError at the individual conversion.

```
result = []
for value in ["2", "x", "4"]:
    try:
        result.append(int(value))
    except ValueError:
        continue
print(result)
```

Expected output

```
[2, 4]
```