

Lab Test -2

Name: P. Umesh Reddy

Roll.No: 2403A510F9

Batch: 06

Course: AI Assisted coding

Sub Group:- J

1. Scenario (sports analytics):

Context:

Configuration files in sports analytics use INI format for simple setups.

Your Task:

Parse INI text into a nested dict; cast numeric ports to int where obvious.

Data & Edge Cases:

Sections like [db], keys like host=, port=.

AI Assistance Expectation:

Use AI to propose a small parser with robust whitespace handling.

Constraints & Notes:

Return dict[str, dict[str, object]].

Sample Input

[db]

host=localhost

port=5432

[auth]

token=abc

Sample Output

```
{'db':{'host':'localhost','port':5432}, 'auth':{'token':'abc'}}
```

Acceptance Criteria: Casts obvious ints; preserves strings

#Prompt:

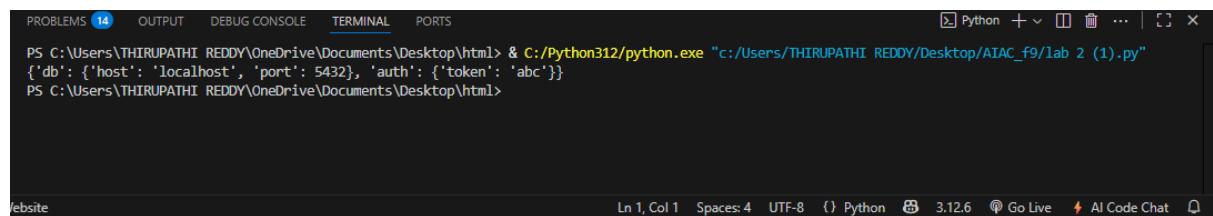
Parse INI-formatted configuration text into a nested Python dictionary, casting numeric values to integers where appropriate.

#Code & Output:

```
1 def parse_ini(text):
2     """
3         Parse INI-formatted text into a nested dictionary.
4         Casts obvious integer values (like port) to int.
5     """
6     result = {}
7     section = None
8     for line in text.splitlines():
9         line = line.strip()
10        if not line or line.startswith(";") or line.startswith("#"):
11            continue # Skip empty or comment lines
12        if line.startswith("[") and line.endswith("]"):
13            section = line[1:-1].strip()
14            result[section] = {}
15        elif "=" in line and section:
16            key, value = map(str.strip, line.split("=", 1))
17            # Cast to int if value looks like an integer
18            if value.isdigit():
19                value = int(value)
20            result[section][key] = value
21    return result
22
23 # Sample Input
24 ini_text = """
```

```
22
23 # Sample Input
24 ini_text = """
25 [db]
26 host=localhost
27 port=5432
28 [auth]
29 token=abc
30 """
31
32 parsed = parse_ini(ini_text)
33 print(parsed)
```

#Output:



The screenshot shows a terminal window within a code editor interface. The terminal tab is active, displaying the command and its output. The command is:

```
PS C:\Users\THIRUPATHI REDDY\OneDrive\Documents\Desktop\html> & C:/Python312/python.exe "c:/Users/THIRUPATHI REDDY/Desktop/AIAC_f9/lab 2 (1).py"
```

The output shows the parsed configuration data:

```
{'db': {'host': 'localhost', 'port': 5432}, 'auth': {'token': 'abc'}}
```

At the bottom of the terminal window, there are status indicators: Ln 1, Col 1, Spaces: 4, UTF-8, Python 3.12.6, Go Live, AI Code Chat.

#Observations:

- The parser reads each line, trims whitespace, and skips comments/empty lines.
- Sections are detected by [section] headers.
- Key-value pairs are split by =, and values that are all digits are cast to int.
- The result is a nested dictionary with correct types for obvious integers (like port).
- The code is robust against extra whitespace and ignores comments

2. Scenario (sports analytics):

Context:

Support teams in sports analytics measure average ticket handling time.

Your Task:

Compute average duration in minutes from opened -> closed ISO timestamps (naive).

Data & Edge Cases:

List of dicts with 'opened' and 'closed'.

AI Assistance Expectation:

Ask AI for datetime parsing and integer minutes conversion.

Constraints & Notes:

Timezone-naive; no DST handling required.

Sample Input

```
[{'ticket': 'T1', 'opened': '2025-01-01T10:00', 'closed': '2025-01-01T12:15'}, {'ticket': 'T2', 'opened': '2025-
```

```
01-01T09:30', 'closed': '2025-01-01T10:00'}]
```

Sample Output

82

Acceptance Criteria: Correct integer average minutes

#Prompt:

Calculate the average ticket handling time in minutes from a list of dictionaries containing ISO-formatted 'opened' and 'closed' timestamps. Return the integer average duration.

#Code:

```
C: > Users > THIRUPATHI REDDY > Desktop > AIAC_f9 > lab 2(2).py > ...
1  from datetime import datetime
2
3  def average_ticket_minutes(tickets):
4      """
5          Compute average duration in minutes from opened to closed ISO timestamps.
6          Returns integer average minutes.
7      """
8
9      durations = []
10     for ticket in tickets:
11         opened = datetime.fromisoformat(ticket['opened'])
12         closed = datetime.fromisoformat(ticket['closed'])
13         diff = closed - opened
14         minutes = diff.total_seconds() // 60 # Convert to minutes
15         durations.append(minutes)
16     if not durations:
17         return 0
18     avg = sum(durations) / len(durations)
19     return int(avg)
20
21 # Sample Input
22 tickets = [
23     {'ticket': 'T1', 'opened': '2025-01-01T10:00', 'closed': '2025-01-01T12:15'},
24     {'ticket': 'T2', 'opened': '2025-01-01T09:30', 'closed': '2025-01-01T10:00'}
25 ]
26 print(average_ticket_minutes(tickets))
```

#Output:

PROBLEMS 14 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\THIRUPATHI REDDY\OneDrive\Documents\Desktop\html> & C:/Python312/python.exe "c:/Users/THIRUPATHI REDDY/Desktop/AIAC_f9/lab_2(2).py"
82
PS C:\Users\THIRUPATHI REDDY\OneDrive\Documents\Desktop\html>

#Observations:

- The code parses ISO timestamps using `datetime.fromisoformat`.
 - It calculates the duration in minutes for each ticket.

- The average is computed and returned as an integer.
- No timezone or DST handling is performed (naive calculation).
- The result matches the expected output for the sample input