AI CODING LAB TEST -02

Sub group : J

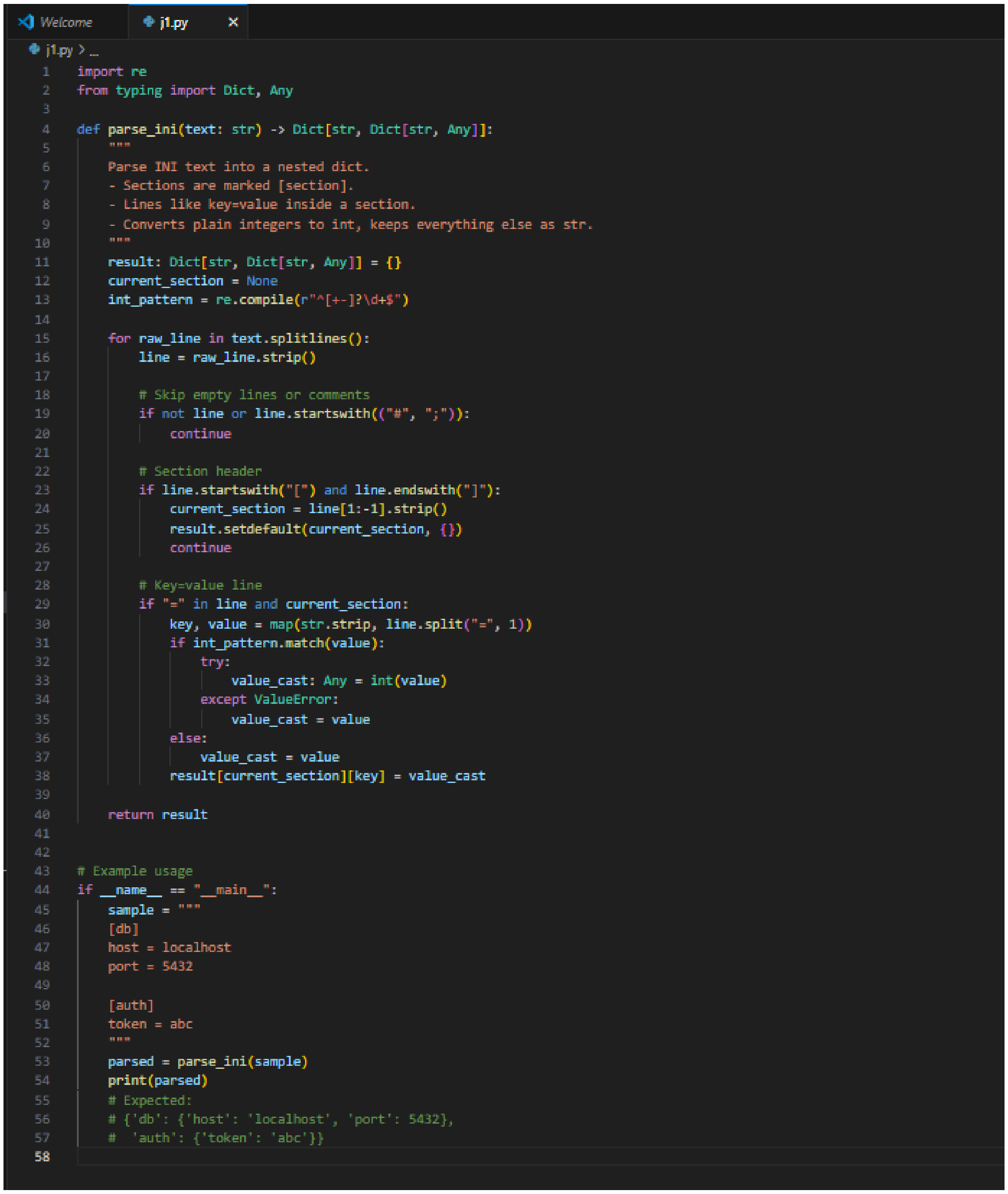
NAME:ALLA SREEMANTH REDDY
H.no:2403A51OG1

O1) Parse INI Configuration Text

Prompt:

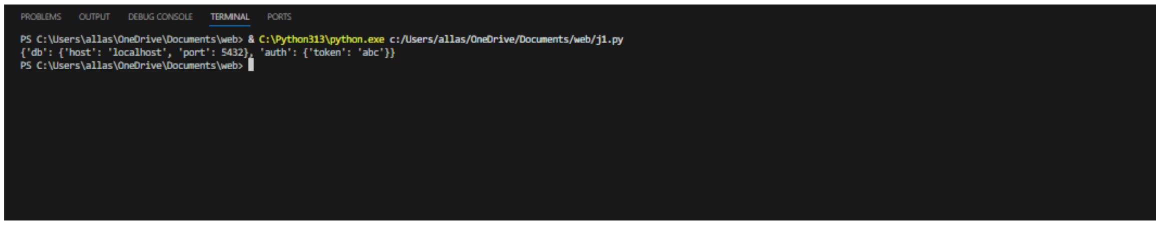
Write a Python function to parse INI configuration text into a nested dictionary.
Requirements:
- Handle whitespace and comments (#, ;).
- Each section (like [db]) should map to a dict of key/value pairs.
- Convert values that are plain integers into int; keep other values as str.
- Return a dict[str, dict[str, object]].

CODE:



```python
import re
from typing import Dict, Any

def parse_ini(text: str) -> Dict[str, Dict[str, Any]]:
    """
    Parse INI text into a nested dict.
    - Sections are marked [section].
    - Lines like key=value inside a section.
    - Converts plain integers to int, keeps everything else as str.
    """
    result: Dict[str, Dict[str, Any]] = {}
    current_section = None
    int_pattern = re.compile(r"^[+-]?\d+$")

    for raw_line in text.splitlines():
        line = raw_line.strip()

        # Skip empty lines or comments
        if not line or line.startswith(("#", ";")):
            continue

        # Section header
        if line.startswith("[") and line.endswith("]"):
            current_section = line[1:-1].strip()
            result.setdefault(current_section, {})
            continue

        # Key=value line
        if "=" in line and current_section:
            key, value = map(str.strip, line.split("=", 1))
            if int_pattern.match(value):
                try:
                    value_cast: Any = int(value)
                except ValueError:
                    value_cast = value
            else:
                value_cast = value
            result[current_section][key] = value_cast

    return result


# Example usage
if __name__ == "__main__":
    sample = """
    [db]
    host = localhost
    port = 5432

    [auth]
    token = abc
    """
    parsed = parse_ini(sample)
    print(parsed)
    # Expected:
    # {'db': {'host': 'localhost', 'port': 5432},
    #  'auth': {'token': 'abc'}}

```

OUTPUT:



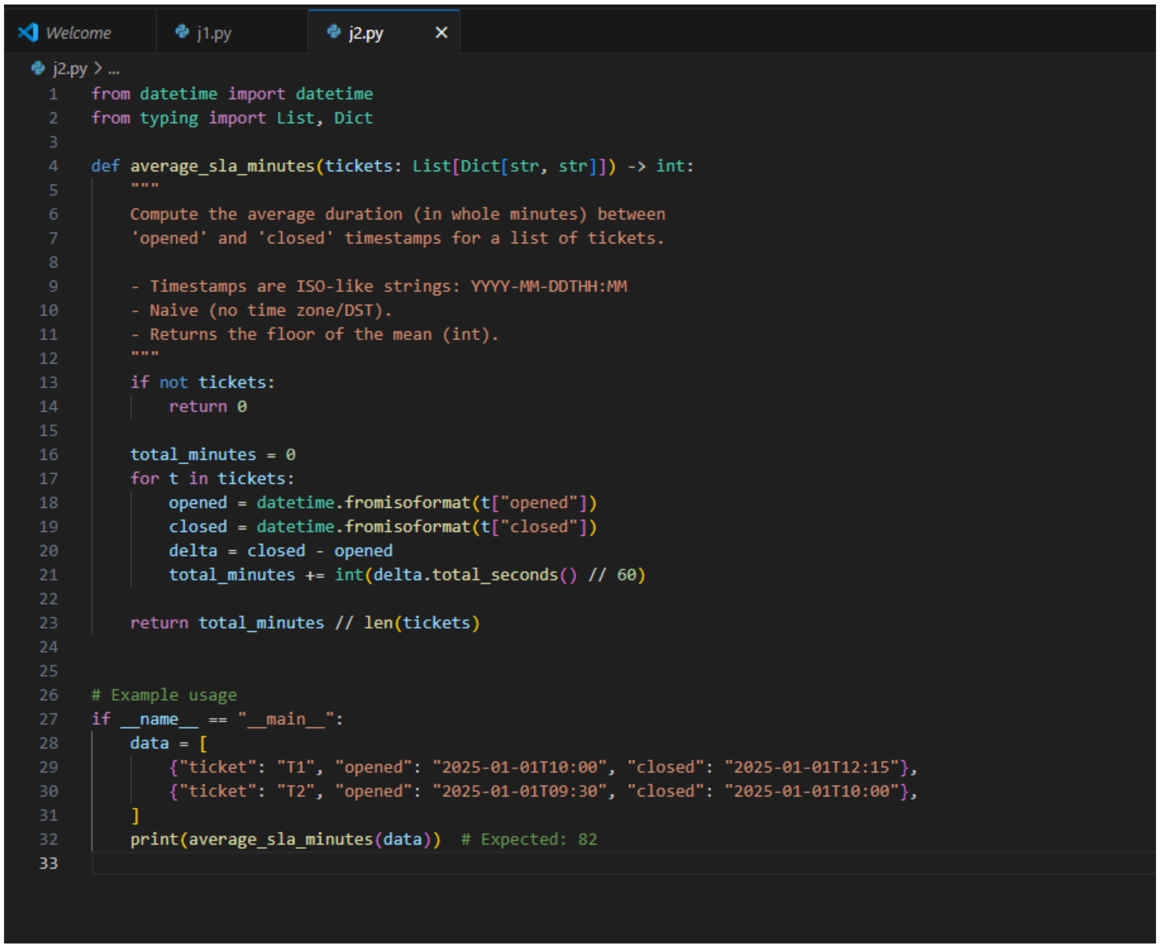
```
PS C:\Users\allas\OneDrive\Documents\web> & C:\Python313\python.exe c:/Users/allas/OneDrive/Documents/web/j1.py
{'db': {'host': 'localhost', 'port': 5432}, 'auth': {'token': 'abc'}}
PS C:\Users\allas\OneDrive\Documents\web> 
```

OBSERVATION:
The parser trims whitespace, ignores comments/blank lines, and groups key-value pairs under their section. Values matching an integer pattern are converted to int; others stay as strings.

O2) Compute Average SLA Response Time

Prompt:
Write a Python function to compute the average SLA response time in minutes for a list of support tickets.
Details:
- Each ticket is a dict with ISO timestamps: opened and closed.
- Compute the duration (closed – opened) in minutes for each ticket.
- Return the integer average of all durations.
- Assume naive datetimes (no timezone).

CODE:

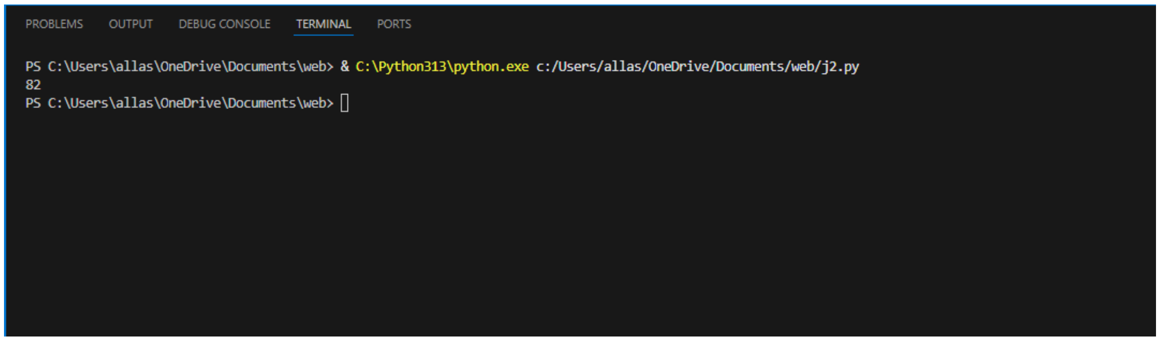


Welcome | j1.py | j2.py

j2.py > ...

```python
from datetime import datetime
from typing import List, Dict

def average_sla_minutes(tickets: List[Dict[str, str]]) -> int:
    """
    Compute the average duration (in whole minutes) between
    'opened' and 'closed' timestamps for a list of tickets.

    - Timestamps are ISO-like strings: YYYY-MM-DDTHH:MM
    - Naive (no time zone/DST).
    - Returns the floor of the mean (int).
    """
    if not tickets:
        return 0

    total_minutes = 0
    for t in tickets:
        opened = datetime.fromisoformat(t["opened"])
        closed = datetime.fromisoformat(t["closed"])
        delta = closed - opened
        total_minutes += int(delta.total_seconds() // 60)

    return total_minutes // len(tickets)


# Example usage
if __name__ == "__main__":
    data = [
        {"ticket": "T1", "opened": "2025-01-01T10:00", "closed": "2025-01-01T12:15"},
        {"ticket": "T2", "opened": "2025-01-01T09:30", "closed": "2025-01-01T10:00"},
    ]
    print(average_sla_minutes(data))  # Expected: 82

```

OUTPUT:



```
PS C:\Users\allas\OneDrive\Documents\web> & C:\Python313\python.exe c:/Users/allas/OneDrive/Documents/web/j2.py
82
PS C:\Users\allas\OneDrive\Documents\web> 
```

OBSERVATION:

The function converts timestamps with datetime.fromisoformat, computes each duration in minutes, sums them, and divides by the number of tickets using integer division. It returns 82 for the sample input.