NAME: SHREYA GOEL
CLASS: 1MCA-B
REGNO: 2347255

PYTHON (Lab Exercise-3)

```python
def decode_encoded_string(stringsplit):
    parts = stringsplit.split('_')

    # Check if the split parts have the required number of elements
    if len(parts) != 3:
        raise ValueError("Invalid encoded string format")

    # Create a dictionary with keys and values
    decoded_dict = {
        "name": parts[0],
        "domain_name": parts[1],
        "register_number": parts[2]
    }

    return decoded_dict

# Example usage:
encoded_string = "Shreya_Job Portal_2347255"
decoded_dict = decode_encoded_string(encoded_string)
print(decoded_dict)
```
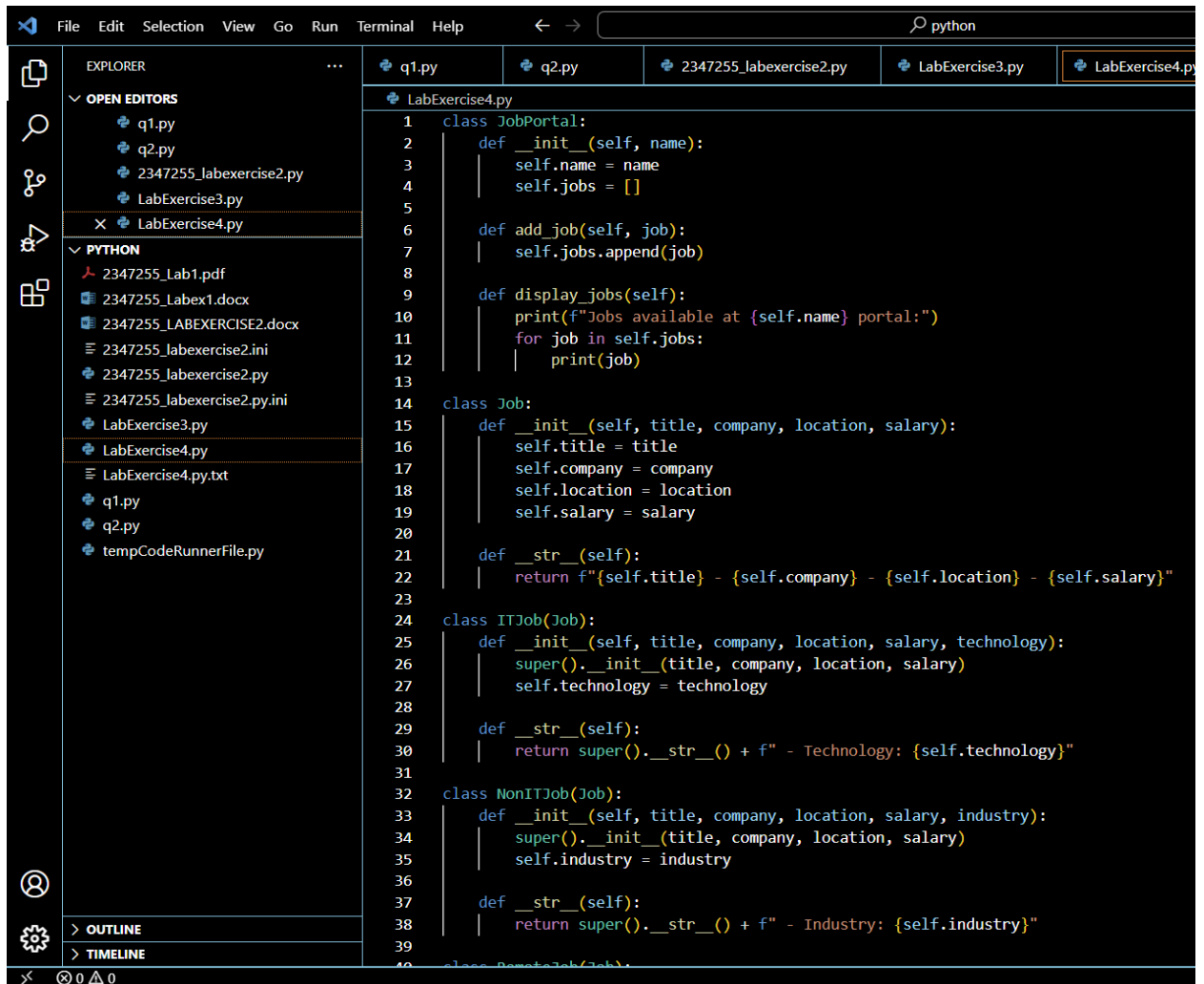
PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**

```
PS C:\python> python -u "c:\python\LabExercise3.py"
{'name': 'Shreya', 'domain_name': 'Job Portal', 'register_number': '2347255'}
PS C:\python> []
```
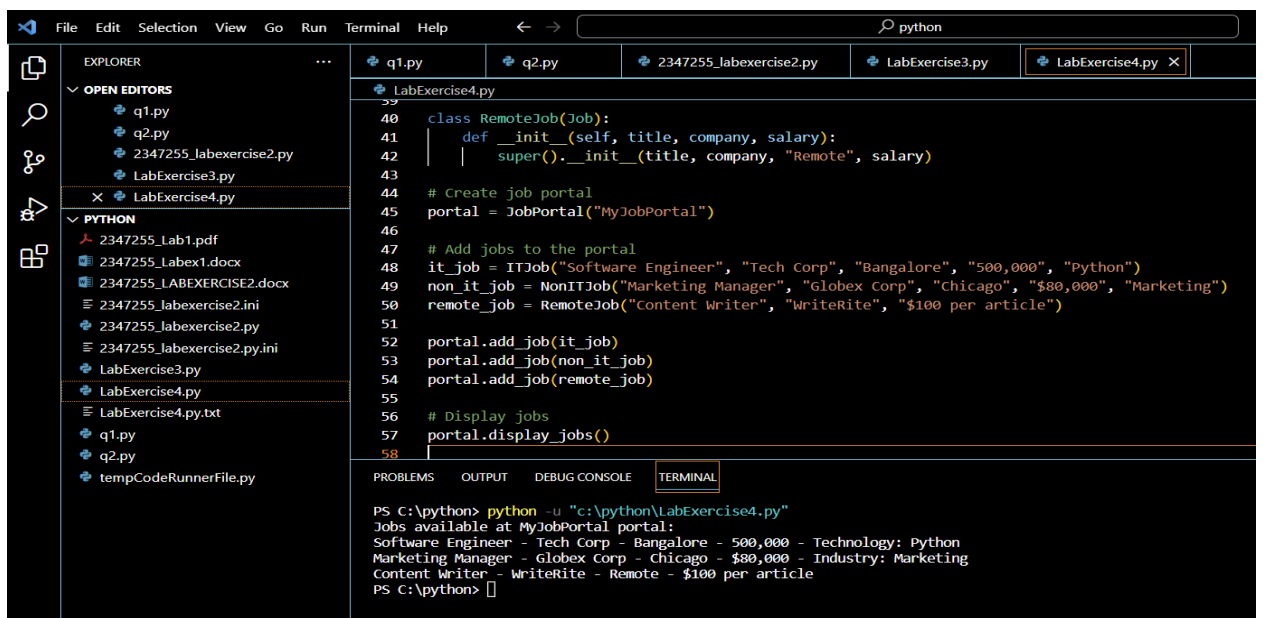
PYTHON (Lab Exercise-4)

```python
class JobPortal:
    def __init__(self, name):
        self.name = name
        self.jobs = []

    def add_job(self, job):
        self.jobs.append(job)

    def display_jobs(self):
        print(f"Jobs available at {self.name} portal:")
        for job in self.jobs:
            print(job)

class Job:
    def __init__(self, title, company, location, salary):
        self.title = title
        self.company = company
        self.location = location
        self.salary = salary

    def __str__(self):
        return f"{self.title} - {self.company} - {self.location} - {self.salary}"

class ITJob(Job):
    def __init__(self, title, company, location, salary, technology):
        super().__init__(title, company, location, salary)
        self.technology = technology

    def __str__(self):
        return super().__str__() + f" - Technology: {self.technology}"

class NonITJob(Job):
    def __init__(self, title, company, location, salary, industry):
        super().__init__(title, company, location, salary)
        self.industry = industry

    def __str__(self):
        return super().__str__() + f" - Industry: {self.industry}"
```

```python
class RemoteJob(Job):
    def __init__(self, title, company, salary):
        super().__init__(title, company, "Remote", salary)

# Create job portal
portal = JobPortal("MyJobPortal")

# Add jobs to the portal
it_job = ITJob("Software Engineer", "Tech Corp", "Bangalore", "500,000", "Python")
non_it_job = NonITJob("Marketing Manager", "Globex Corp", "Chicago", "$80,000", "Marketing")
remote_job = RemoteJob("Content Writer", "WriteRite", "$100 per article")

portal.add_job(it_job)
portal.add_job(non_it_job)
portal.add_job(remote_job)

# Display jobs
portal.display_jobs()
```

```
PS C:\python> python -u "c:\python\LabExercise4.py"
Jobs available at MyJobPortal portal:
Software Engineer - Tech Corp - Bangalore - 500,000 - Technology: Python
Marketing Manager - Globex Corp - Chicago - $80,000 - Industry: Marketing
Content Writer - WriteRite - Remote - $100 per article
PS C:\python>
```