22MCA0139

Rajat Singh

Python Programming LAB

Assessment 3

1) Write a Python program to find date information present in the given sentience using predefined string handling functions.

Constraint: The date can be in the following format. 25/03/2020 25-03-2020 25.03.2020 25/3/20 5-03-20 05.3.20

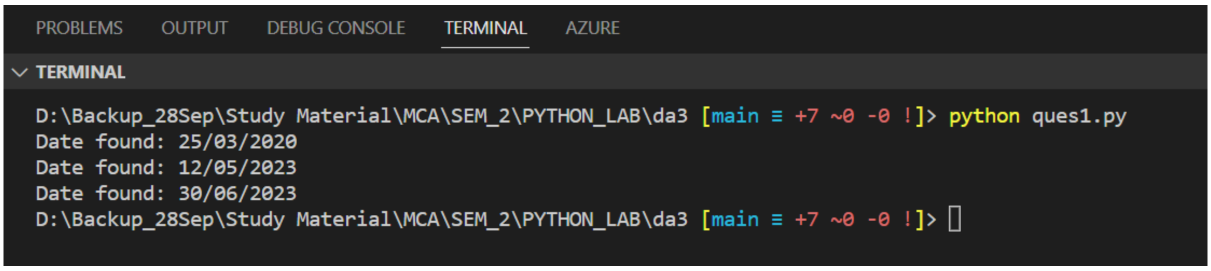
The data can be present in any place in the entire sentence.

Solution:

```python
def extract_date(sentence):
    date_separators = ['/', '-', '.']
    formats = ['%d/%m/%Y', '%d-%m-%Y', '%d.%m.%Y', '%d/%m/%y', '%d-%m-%y',
'%d.%m.%y']

    sentence = sentence.replace('.', '')

    words = sentence.split()
    for word in words:
        for separator in date_separators:
            if separator in word:
                parts = word.split(separator)
                if len(parts) == 3:
                    for date_format in formats:
                        try:
                            date = datetime.datetime.strptime(word,
date_format).date()
                            formatted_date = date.strftime('%d/%m/%Y')
                            return formatted_date
                        except ValueError:
                            pass

    return None

# Example usage
import datetime

sentence = "The event will take place on 25/03/2020. Don't forget to mark your
calendar!"
```

```python
Input=["The project deadline is set for 25-03-2020. Make sure to submit your
work on time!",
"The release date of the highly anticipated movie is 12/05/2023. Get ready for
an exciting cinematic experience!",
"The registration deadline for the competition is 30-06-23. Don't miss your
chance to participate."]
for i in Input:
    date = extract_date(i)
    if date:
        print(f"Date found: {date}")
    else:
        print("No date found in the sentence.")
```

Output:



```
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]> python ques1.py
Date found: 25/03/2020
Date found: 12/05/2023
Date found: 30/06/2023
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]> 
```

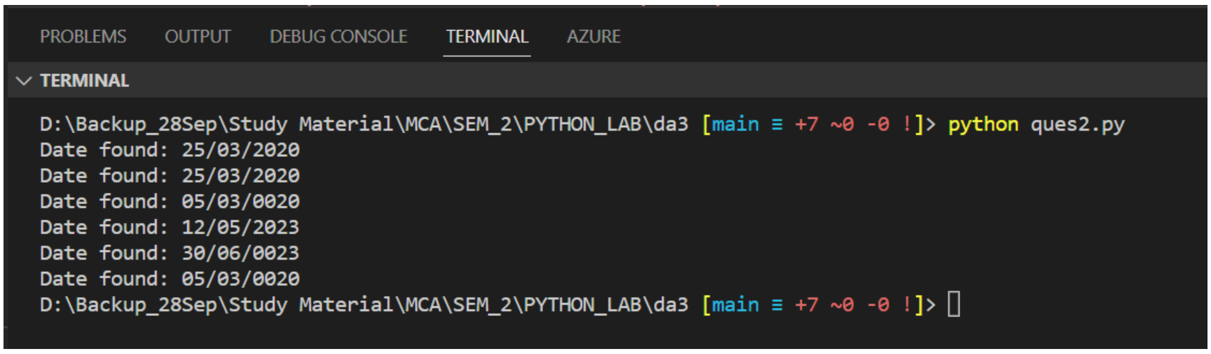
2) Solve task-1 using regular expression instead of using string handling functions.

Solution:

```python
import re

def extract_date(sentence):
    date_regex = r'\b(\d{1,2})[-./](\d{1,2})[-./](\d{2}|\d{4})\b'
    matches = re.findall(date_regex, sentence)
    if matches:
        for match in matches:
            day = match[0]
            month = match[1]
            year = match[2]
            formatted_date =
f'{day.zfill(2)}/{month.zfill(2)}/{year.zfill(4)}'
            return formatted_date
    return None

sentence = "The event will take place on 25/03/2020. Don't forget to mark your
calendar!"
```

```
Input=["The project deadline is set for 25-03-2020. Make sure to submit your
work on time!",
"Join us for a special event on 25.03.2020. Save the date!",
"The workshop will be held on 05.3.20. Register in advance to secure your
spot.",
"The release date of the highly anticipated movie is 12/05/2023. Get ready for
an exciting cinematic experience!",
"The registration deadline for the competition is 30-06-23. Don't miss your
chance to participate.",
'05.3.20']
for i in Input:
    date = extract_date(i)
    if date:
        print(f"Date found: {date}")
    else:
        print("No date found in the sentence.")
```

Output:



```
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]> python ques2.py
Date found: 25/03/2020
Date found: 25/03/2020
Date found: 05/03/0020
Date found: 12/05/2023
Date found: 30/06/0023
Date found: 05/03/0020
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]>
```

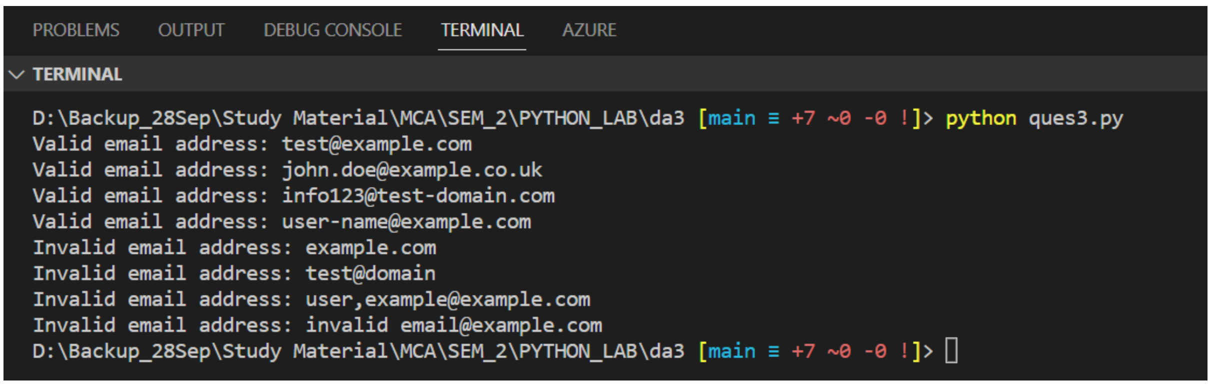
3) Check whether the input email address is valid or not using predefined string handling functions.

Solution:

```
def is_valid_email(email):
    if '@' not in email:
        return False

    username, domain = email.split('@')

    if not username:
        return False

    allowed_username_chars = set('.-
_abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789')
```

```python
    if any(char not in allowed_username_chars for char in username):
        return False

    domain_parts = domain.split('.')
    if len(domain_parts) < 2:
        return False

    tld = domain_parts[-1]
    if not tld.isalpha() or len(tld) < 2:
        return False

    return True

test_cases = [
    "test@example.com",
    "john.doe@example.co.uk",
    "info123@test-domain.com",
    "user-name@example.com",
    "example.com",
    "test@domain",
    "user,example@example.com",
    "invalid email@example.com"
]

# Test the email addresses
for email in test_cases:
    if is_valid_email(email):
        print(f"Valid email address: {email}")
    else:
        print(f"Invalid email address: {email}")
```

Output:



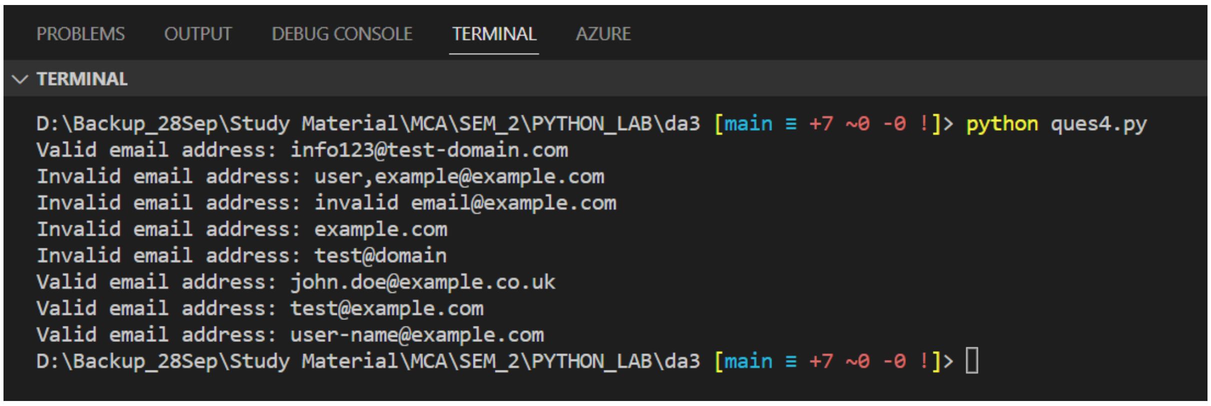
```
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]> python ques3.py
Valid email address: test@example.com
Valid email address: john.doe@example.co.uk
Valid email address: info123@test-domain.com
Valid email address: user-name@example.com
Invalid email address: example.com
Invalid email address: test@domain
Invalid email address: user,example@example.com
Invalid email address: invalid email@example.com
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]>
```

4) Solve task-3 using regular expression instead of using string handling functions.

Solution:

```python
import re

def is_valid_email(email):
    pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'
    return re.match(pattern, email) is not None

test_cases = [
    "info123@test-domain.com",
    "user,example@example.com",
    "invalid email@example.com",
    "example.com",
    "test@domain",
    "john.doe@example.co.uk",
    "test@example.com",
    "user-name@example.com"
]

for email in test_cases:
    if is_valid_email(email):
        print(f"Valid email address: {email}")
    else:
        print(f"Invalid email address: {email}")
```

Output:



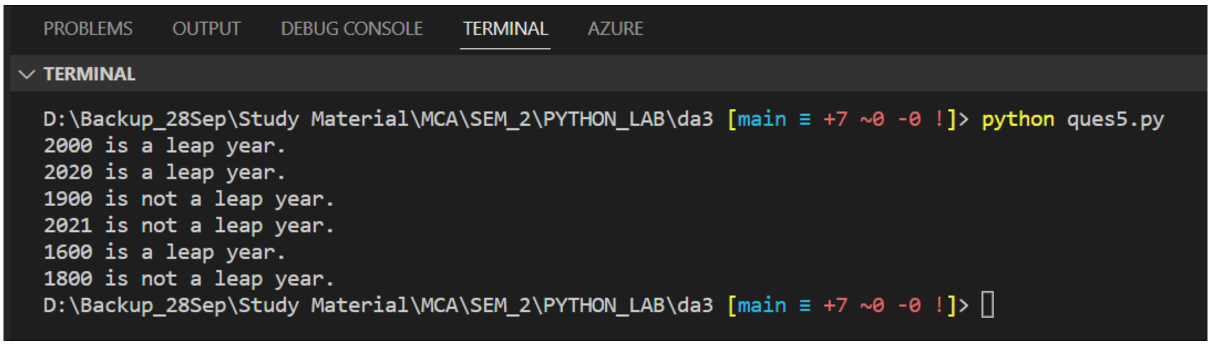
```
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]> python ques4.py
Valid email address: info123@test-domain.com
Invalid email address: user,example@example.com
Invalid email address: invalid email@example.com
Invalid email address: example.com
Invalid email address: test@domain
Valid email address: john.doe@example.co.uk
Valid email address: test@example.com
Valid email address: user-name@example.com
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]> 
```

5) Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean True, otherwise return False. Create a user-defined function to solve this leap-year task.

Solution:

```python
def is_leap_year(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                return True  # Divisible by 400, leap year
            else:
                return False  # Divisible by 100 but not 400, not a leap year
        else:
            return True  # Divisible by 4 but not 100, leap year
    else:
        return False  # Not divisible by 4, not a leap year

test_cases = [
    2000,
    2020,
    1900,
    2021,
    1600,
    1800
]

for year in test_cases:
    if is_leap_year(year):
        print(f"{year} is a leap year.")
    else:
        print(f"{year} is not a leap year.")
```

Output:



```
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]> python ques5.py
2000 is a leap year.
2020 is a leap year.
1900 is not a leap year.
2021 is not a leap year.
1600 is a leap year.
1800 is not a leap year.
D:\Backup_28Sep\Study Material\MCA\SEM_2\PYTHON_LAB\da3 [main ≡ +7 ~0 -0 !]>
```