Day2-OOPS

Question 1 – JSONPlaceholder Users → Class Objects

Create a class User that stores data fetched from https://jsonplaceholder.typicode.com/users.

Requirements:

1. The constructor (__init__) should accept id, name, and email.

2. Define a method showUser() to print the user details in a readable format.

o Example: User #1 → Leanne Graham (Sincere@april.biz)

3. Define another method getEmailDomain() to return only the domain part of the email.

o Example: april.biz

4. Fetch users from the API and create multiple User objects.

5. Print details of at least 5 users using showUser() and getEmailDomain().

Code:

```
import requests
apiURL="https://jsonplaceholder.typicode.com/users"
response = requests.get(apiURL)
print(response)
class jsonPlaceholderUser:
    def __init__(self,i,n,e):
        self.id=i
        self.name=n
        self.email=e
    def showUser(self):
        print(f"{self.name} ( {self.email})")
    def getEmailDomain(self):
        return self.email.split('@')[-1]

if response.status_code == 200:
```

```
    data=response.json()
    users=[]
    for i in data:
        id=i["id"]
        name=i["name"]
        email=i["email"]
        # abc=jsonPlaceholderUser(id,name,email)
        # abc.showUser()
        users.append(jsonPlaceholderUser(id,name,email))


    for u in users[:5]:
        u.showUser()
        print("Email domain:",u.getEmailDomain())
```

output:

```
python day2taskq1.py
<Response [200]>
Leanne Graham ( Sincere@april.biz)
Email domain: april.biz
Ervin Howell ( Shanna@melissa.tv)
Email domain: melissa.tv
Clementine Bauch ( Nathan@yesenia.net)
Email domain: yesenia.net
Patricia Lebsack ( Julianne.OConner@kory.org)
Email domain: kory.org
Chelsey Dietrich ( Lucio_Hettinger@annie.ca)
Email domain: annie.ca
```

Question 2 – JSONPlaceholder Posts → Blog System

Create a class Post to manage blog posts fetched from https://jsonplaceholder.typicode.com/posts.

Requirements:

1. The constructor should accept userId, id, title, and body.

2. Define a method showSummary() that prints the post ID and first 20 characters of the title.

o Example: Post #1 → sunt aut facere repellat...

3. Define another method getWordCount() that counts how many words are in the body.

4. Fetch posts from the API and create multiple Post objects.

5. For the first 3 posts, show summary and word count.

Code:

```
import requests

apiURL="https://jsonplaceholder.typicode.com/posts"

response = requests.get(apiURL)


class Post:
    def __init__(self,u,i,t,b):
        self.userid=u
        self.id=i
        self.title=t
        self.body=b

    def showSummary(self):
        print(f"{self.id}-{self.title[:20]}")

    def getWordCount(self):
        return len(self.body.split())
```

```
if response.status_code == 200:
    data=response.json()
    posts=[]
    for p in data:
        post = Post(p["userId"], p["id"], p["title"], p["body"])
        posts.append(post)
    for post in posts[:3]:
        post.showSummary()
        print("Word Count:", post.getWordCount())
        print("-" * 40)
else:
    print("Failed to fetch posts:", response.status_code)
```

Output:

1-sunt aut facere repe

Word Count: 23

2-qui est esse

Word Count: 31

3-ea molestias quasi e

Word Count: 26

Question 3 – DummyJSON Products → Flipkart Style

Use API: https://dummyjson.com/products

Create a class Product to store details of each product.

Requirements:

1. The constructor should accept id, title, price, and stock.

2. Define a method showDetails() to display product info.

o Example: Laptop (₹50000) – Stock: 12

3. Define another method buyProduct(qty) that reduces stock if enough stock is available,

otherwise print "Out of stock".

4. Fetch product data from API and create multiple Product objects.

5. Simulate a shopping cart by buying:

o 2 units of the first product.

o 1 unit of the second product.

o Try buying more than stock for the third product.

Code:

```
import requests

apiURL="https://dummyjson.com/products"

response=requests.get(apiURL)


class Product:
    def __init__(self, pid, title, price, stock):
        self.id = pid
        self.title = title
        self.price = price
        self.stock = stock


    def showDetails(self):
        print(f"{self.title} (₹{self.price}) Stock: {self.stock}")


    def buyProduct(self, qty):
        if self.stock >= qty:
            self.stock -= qty
            print(f" Bought {qty} unit(s) of {self.title}. Remaining stock: {self.stock}")
        else:
```

```
        print(f" Out of stock: {self.title}. Available: {self.stock}, Requested: {qty}")


if response.status_code == 200:
    data = response.json()
    products = []
    # Create Product objects
    for p in data["products"]:
        prod = Product(p["id"], p["title"], p["price"], p["stock"])
        products.append(prod)
    # Show first 3 products
    print("\n--- Product Catalog (First 3) ---")
    for prod in products[:3]:
        prod.showDetails()

    print("\n--- Shopping Simulation ---")
    # Buy 2 units of first product
    products[0].buyProduct(2)

    # Buy 1 unit of second product
    products[1].buyProduct(1)

    # Try to buy more than stock of third product
    products[2].buyProduct(products[2].stock + 5)
else:
    print("Failed to fetch products:", response.status_code)
```

Output:

--- Product Catalog (First 3) ---

Essence Mascara Lash Princess (₹9.99) Stock: 99

Eyeshadow Palette with Mirror (₹19.99) Stock: 34

Powder Canister (₹14.99) Stock: 89

--- Shopping Simulation ---

Bought 2 unit(s) of Essence Mascara Lash Princess. Remaining stock: 97

Bought 1 unit(s) of Eyeshadow Palette with Mirror. Remaining stock: 33

Out of stock: Powder Canister. Available: 89, Requested: 94