

Name: Muhammad Abdullah

Roll no: 016

Section: 3A

Department: BSDSM

Task #1: Artificial Intelligence Lab

Q#1 Create a class User that keeps track of how many user objects have been created.

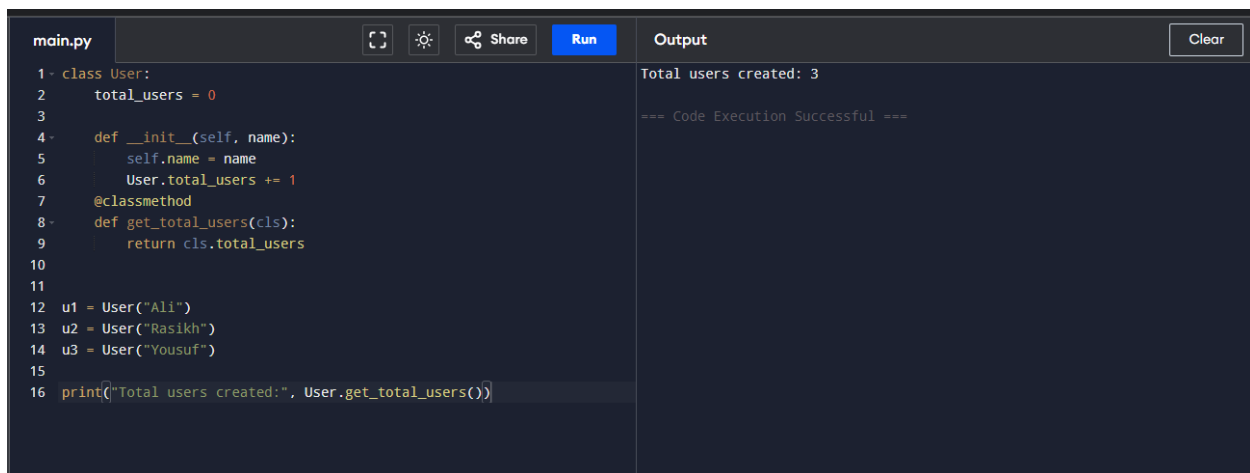
Requirements:

- Use a class variable `total_users`
- Use a class method `get_total_users()` to return the total number of users
- Increment the counter every time a new object is created

Task:

Write a program that:

1. Creates at least 3 user objects
2. Prints the total number of users using the class method



The screenshot shows a code editor with a file named `main.py`. The code defines a `User` class with a class variable `total_users` initialized to 0. The `__init__` method increments `total_users` by 1 for each new instance. A class method `get_total_users` returns the current value of `total_users`. Three `User` objects are created with names "Ali", "Rasikh", and "Yousuf". Finally, the program prints the total number of users created, which is 3. The output panel on the right shows the result of the execution.

```
1- class User:
2-     total_users = 0
3-
4-     def __init__(self, name):
5-         self.name = name
6-         User.total_users += 1
7-
8-     @classmethod
9-     def get_total_users(cls):
10-         return cls.total_users
11-
12- u1 = User("Ali")
13- u2 = User("Rasikh")
14- u3 = User("Yousuf")
15-
16- print("Total users created:", User.get_total_users())
```

Output

Total users created: 3

=== Code Execution Successful ===

Q2. Create a class Product with:

- A class variable `tax_rate = 0.15`

Requirements:

- Use a class method `update_tax_rate(new_rate)` to update the tax rate
- Use a static method `calculate_tax(price)` that returns the tax for a given price
- Demonstrate that changing the tax rate affects all products

main.py	Output
<pre>1- class Product: 2- tax_rate = 0.15 3- def __init__(self, name, price): 4- self.name = name 5- self.price = price 6- 7- @classmethod 8- def update_tax_rate(cls, new_rate): 9- cls.tax_rate = new_rate 10- 11- @staticmethod 12- def calculate_tax(price): 13- return price * Product.tax_rate 14- 15- p1 = Product("Laptop", 1000) 16- p2 = Product("Phone", 500) 17- 18- print("Old Tax Rate:", Product.tax_rate) 19- print("Tax on Laptop:", Product.calculate_tax(p1.price)) 20- print("Tax on Phone:", Product.calculate_tax(p2.price)) 21- 22- Product.update_tax_rate(0.20) 23- 24- print("\nNew Tax Rate:", Product.tax_rate) 25- print("Tax on Laptop:", Product.calculate_tax(p1.price)) 26- print("Tax on Phone:", Product.calculate_tax(p2.price))</pre>	<pre>Old Tax Rate: 0.15 Tax on Laptop: 150.0 Tax on Phone: 75.0 New Tax Rate: 0.2 Tax on Laptop: 200.0 Tax on Phone: 100.0 === Code Execution Successful ===</pre>

Q3. Create a class Validator.

Requirements:

- Write a static method `is_valid_email(email)` that:
 - Returns True if the email contains @ and .
 - Otherwise returns False
- Do NOT use any instance or class variables

Task:

Test the method with at least 3 email addresses.

main.py	Share	Run	Output
<pre>1 class Validator: 2 3 @staticmethod 4 def is_valid_email(email): 5 if "@" in email and "." in email: 6 return True 7 else: 8 return False 9 10 print(Validator.is_valid_email("rasikh@gmail.com")) 11 print(Validator.is_valid_email("ali@yahoo")) 12 print(Validator.is_valid_email("jefferyepstein@domain.org"))</pre>			<pre>True False True === Code Execution Successful ===</pre>