

Assesment:

Solution 1: In Python, when you assign `arr2=arr1` we are not creating a new list; we are making `arr2` reference the same list object `arr1`. It means that `arr2` and `arr1` point to the same list, so we are changing one that reflects in another. `arr3=arr1[:]` creates a new list `arr3` this is a shallow copy `arr1`. This means that `arr3` is a new list object containing the same elements `arr1`, but it is a separate list. Changes made in `arr3` will not effect `arr1`.

Therefore, `arr2==arr3` will be True because both `arr2` and `arr3` contain the same elements, even though they are different objects in memory.

Solution 2:

To hide class variables from being accessed outside the class, we have to use the concept of encapsulation in OOPS. In Python we will use private variables. Private variables are initialized by double underscores.

```
class Calc():
```

```
    def __init__(self):
```

```
        self.__result = 0
```

```
    def add(self, x, y):
```

```
        self.__result = x + y
```

```
    def get_result(self):
```

```
        return self.__result
```

```
c = Calc()
```

```
c.add(2, 3)
```

```
print(c.get_result())
```

`print(calc.__result)` this will raise an attribute error because we are trying to print the `__result`

```
pandaspro.py x oopspro.py x calcpro.py x basicquestion.py x
1 class Calc():
2     def __init__(self):
3         self.__result = 0
4     def add(self, x, y):
5         self.__result = x + y
6     def get_result(self):
7         return self.__result
8 c = Calc()
9 c.add(2, 3)
10 print(c.get_result())
11 print(Calc.__result)
```

Run: calcpro x

C:\Users\Welcome\AppData\Local\Programs\Python\Python311\python.exe C:\Users\Welcome\PycharmProjects\pyt
5
Traceback (most recent call last):
File "C:\Users\Welcome\PycharmProjects\pythonProgram\calcpro.py", line 11, in <module>
 print(Calc.__result)
 ^^^^^^^^^^^^^
AttributeError: type object 'Calc' has no attribute '__result'. Did you mean: 'get_result'?

Process finished with exit code 1

Solution3:

1:prepare PostgreSQL database

2:Initial Data migration.

3:Enable Replication

4:Incremental Data synchronization

5:switch application to PostgreSQL

6:final data synchronization

7:Decommission mysql

Solution 4:

Task1:

User profile image generation api:

1: We have to use PIL library to generate user profile image based on the user name and gender

2: Save the generated image

Install pil library:

#Code for the image to generate

```
from PIL import Image, ImageDraw, ImageFont;
```

```
import os
```

```
def generate_profile_image(username, gender):
```

```
    # Assuming you have a directory named 'profile_images' to save the images
```

```
    os.makedirs('profile_images', exist_ok=True)
```

```
    filename = f"profile_images/{username}.png"
```

```
    image = Image.new('RGB', (200, 200), color='white')
```

```
    draw = ImageDraw.Draw(image)
```

```
    font = ImageFont.load_default()
```

```
    Generate image based on gender
```

```
    if gender.lower() == 'male':
```

```
        color = 'green'
```

```
    else:
```

```
        color = 'pink'
```

```
    draw.text((50, 50), username, fill=color, font=font)
```

```
    image.save(filename)
```

```
    return filename
```

Task 2:

2 :User Search Api:

```
from django.http import JsonResponse
```

```
from django.views.decorators.http import require_HTTP_methods
```

```
from .models import User
```

```
@require_http_methods(["GET"])
```

```
def search_users(request):
```

```
    name = request.GET.get('name')
```

```
    email = request.GET.get('email')
```

```
    location = request.GET.get('location')
```

Assuming that we have fields 'name', 'email', and 'location' in our User model

```
    users = User.objects.filter(name__icontains=name, email__icontains=email,  
                                location__icontains=location)
```

```
    user_data = [{ 'name': user.name, 'email': user.email, 'location': user.location } for user in users]
```

```
    return JsonResponse({'users': user_data})
```

Created URL :

```
from django.urls import path;
```

```
from . import views
```

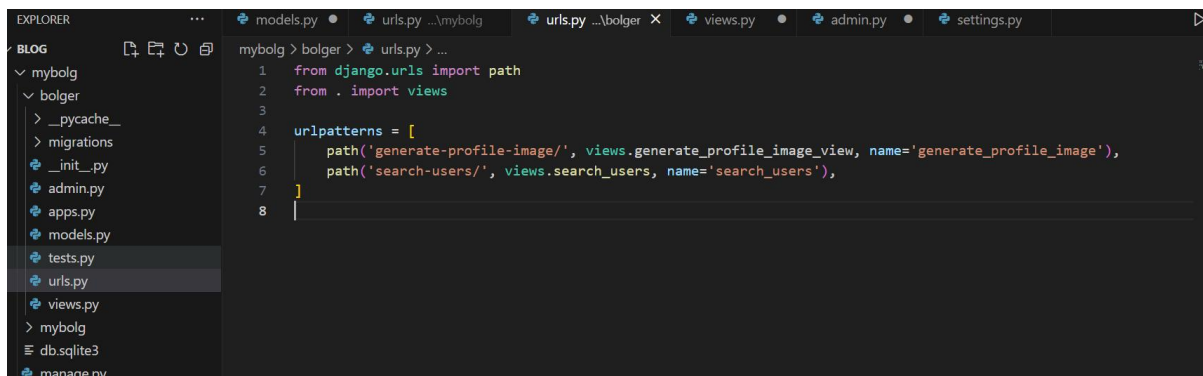
```
urlpatterns = [
```

```
    path('generate-profile-image/', views.generate_profile_image_view,  
        name='generate_profile_image'),
```

```
    path('search-users/', views.search_users, name='search_users'),
```

```
]
```

Hi sir I am adding here screenshot of VS Code please check it for better understanding:



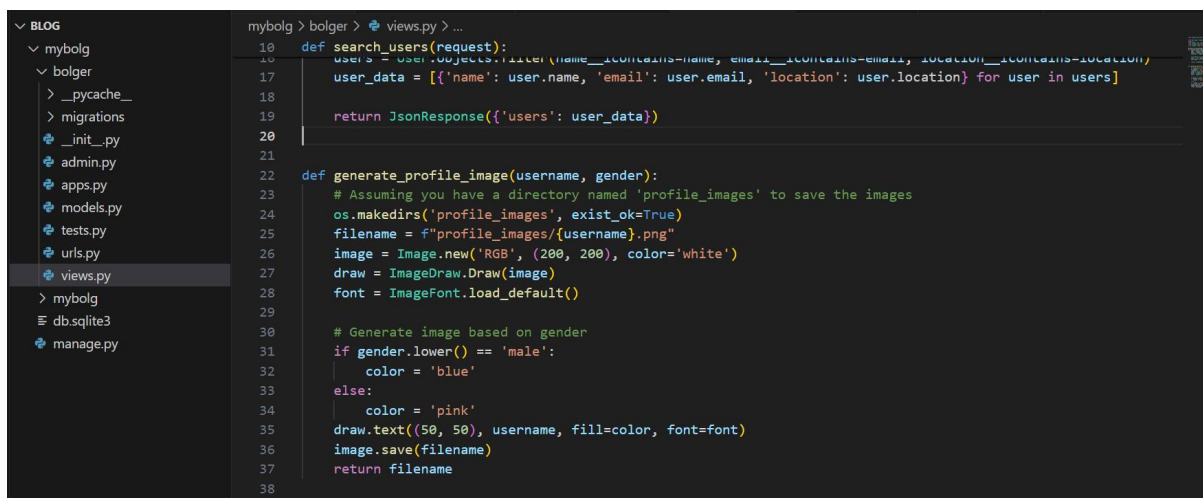
EXPLORER

- BLOG
 - mybolg
 - bolger
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - mybolg
 - db.sqlite3
 - manage.py

mybolg > bolger > urls.py > ...

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('generate-profile-image/', views.generate_profile_image_view, name='generate_profile_image'),
6     path('search-users/', views.search_users, name='search_users'),
7 ]
8
```

```
1 from django.shortcuts import render
2
3 from PIL import Image, ImageDraw, ImageFont
4 import os
5 from django.http import JsonResponse
6 from django.views.decorators.http import require_http_methods
7 from .models import User # Assuming you have a User model
8
9 @require_http_methods(["GET"])
10 def search_users(request):
11     name = request.GET.get('name')
12     email = request.GET.get('email')
13     location = request.GET.get('location')
14
15     # Assuming you have fields 'name', 'email', and 'location' in your User model
16     users = User.objects.filter(name__icontains=name, email__icontains=email, location__icontains=location)
```



BLOG

- mybolg
 - bolger
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - urls.py
 - views.py
 - mybolg
 - db.sqlite3
 - manage.py

mybolg > bolger > views.py > ...

```
10 def search_users(request):
11     users = User.objects.filter(name__icontains=name, email__icontains=email, location__icontains=location)
12     user_data = [{'name': user.name, 'email': user.email, 'location': user.location} for user in users]
13
14     return JsonResponse({'users': user_data})
15
16
17
18
19
20
21
22 def generate_profile_image(username, gender):
23     # Assuming you have a directory named 'profile_images' to save the images
24     os.makedirs('profile_images', exist_ok=True)
25     filename = f"profile_images/{username}.png"
26     image = Image.new('RGB', (200, 200), color='white')
27     draw = ImageDraw.Draw(image)
28     font = ImageFont.load_default()
29
30     # Generate image based on gender
31     if gender.lower() == 'male':
32         color = 'blue'
33     else:
34         color = 'pink'
35     draw.text((50, 50), username, fill=color, font=font)
36     image.save(filename)
37     return filename
38
```