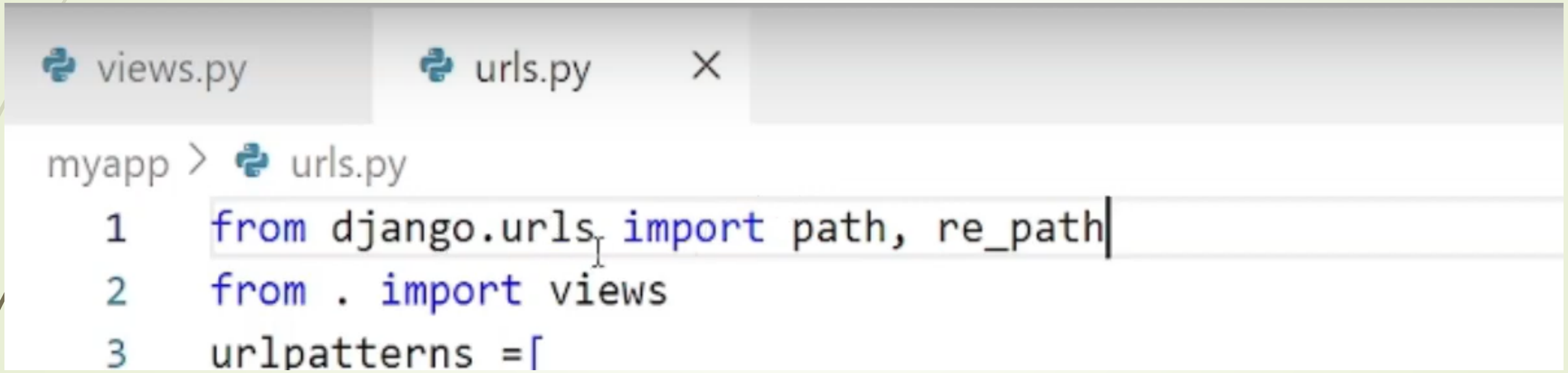
Regular Expression in URLs

BY Kamalpreet Kaur

Regular expressions in Django

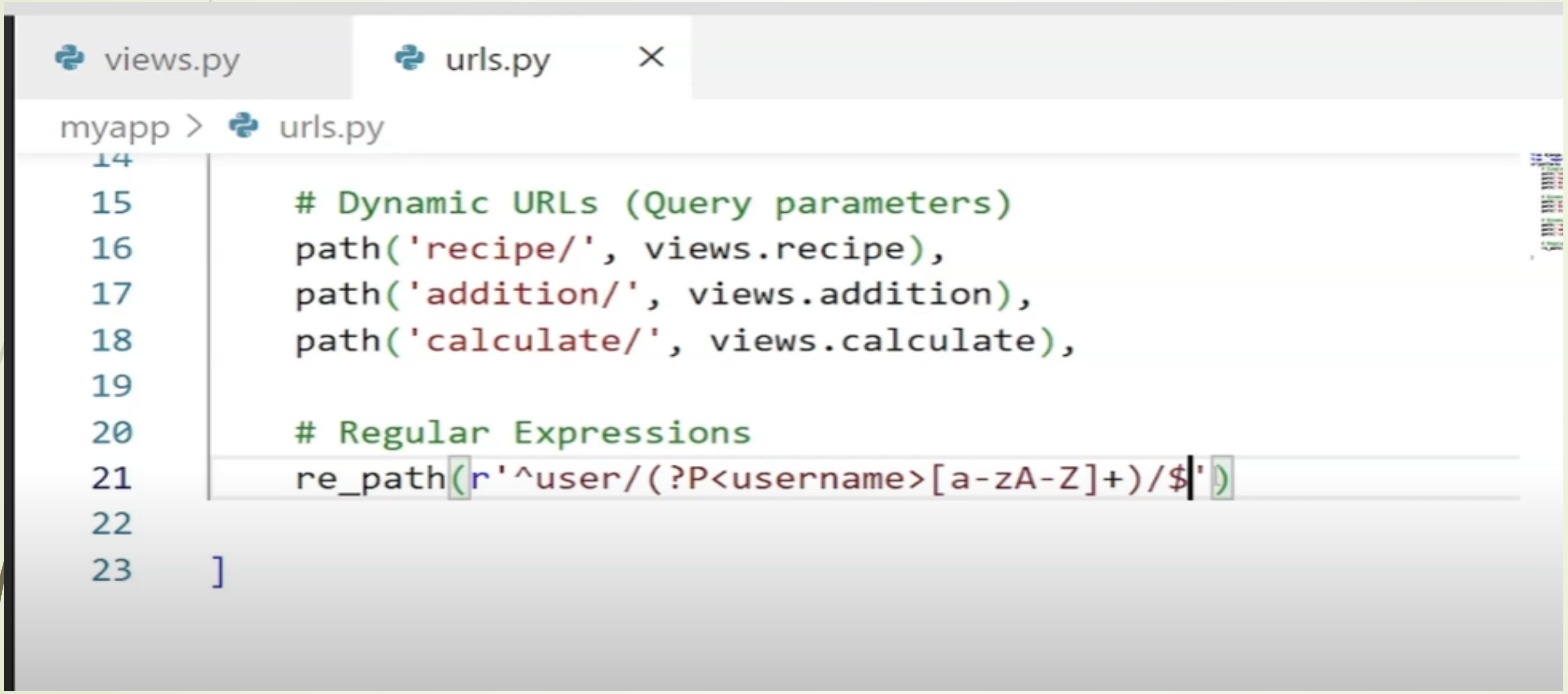
- ^ (caret)
- $(dollar)
- /(forward slash)
- ()(parentheses)
- ?P<name>
- \d
- \s
- [0-9]
- [a-zA-Z]
- [\w]
- [-]
- +
- *

Importing the Path in Url.py



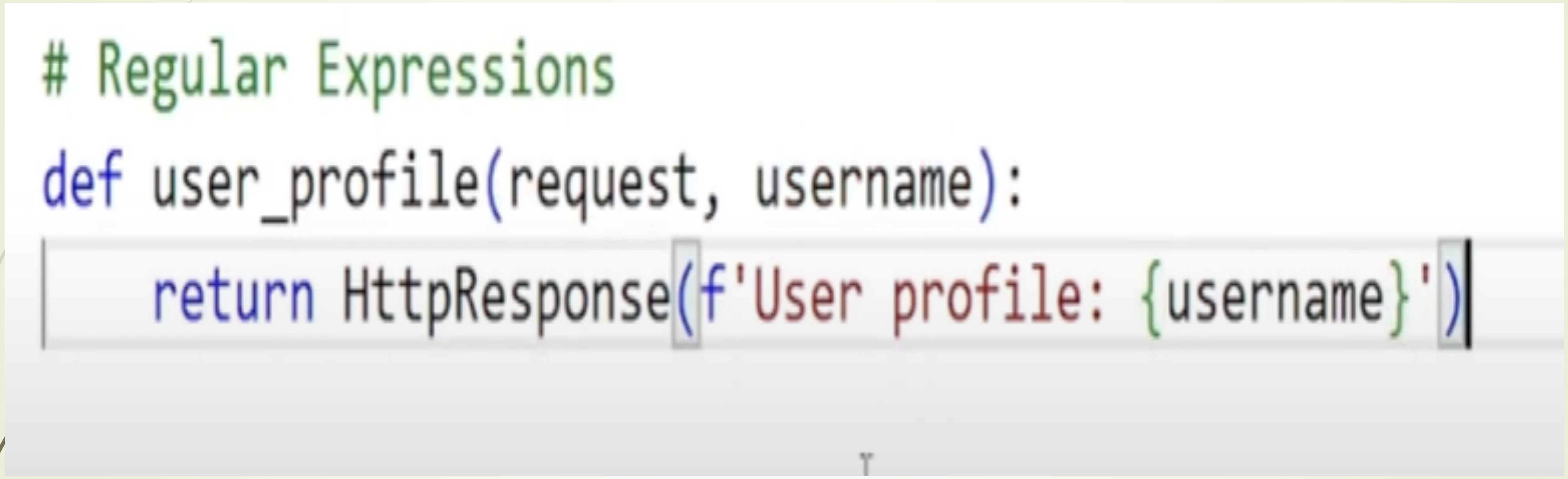
```python
from django.urls import path, re_path
from . import views
urlpatterns =[
```

Defining a path



```
    # Dynamic URLs (Query parameters)
    path('recipe/', views.recipe),
    path('addition/', views.addition),
    path('calculate/', views.calculate),

    # Regular Expressions
    re_path(r'^user/(?P<username>[a-zA-Z]+)/$')

]
```

Function in view.py

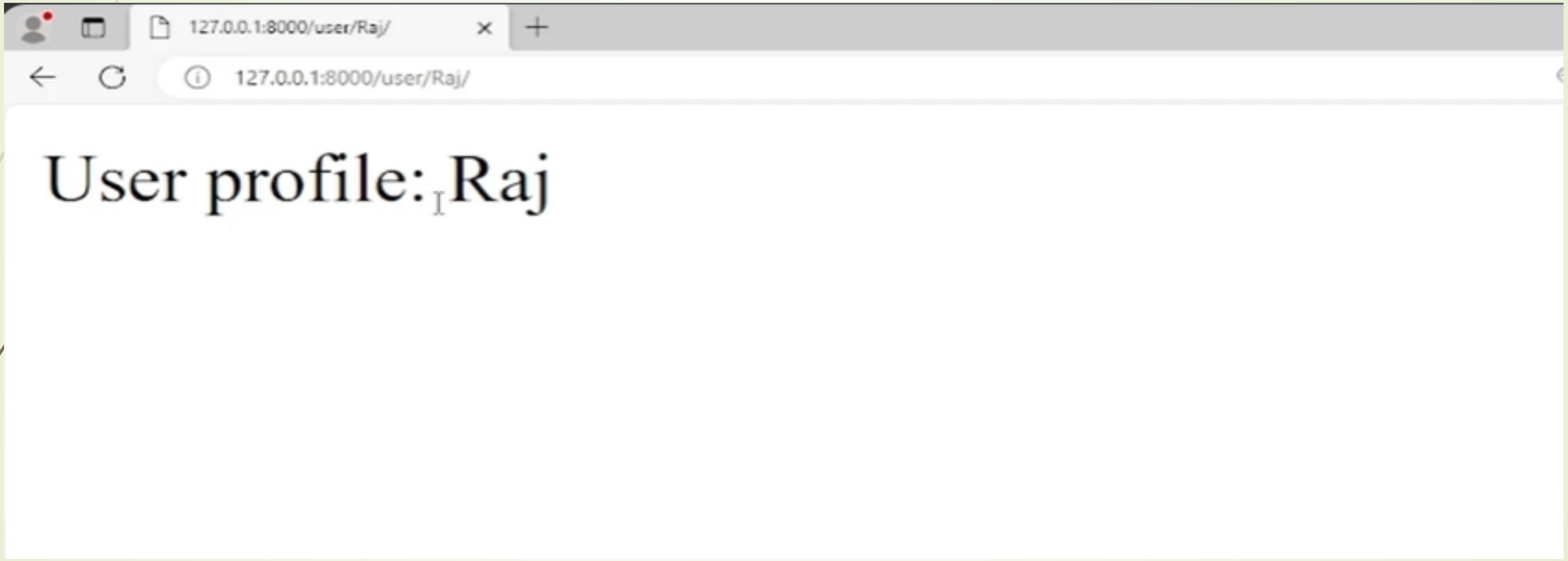


```
# Regular Expressions
def user_profile(request, username):
    return HttpResponse(f'User profile: {username}')
```

•

- After running the server
- Copy the link

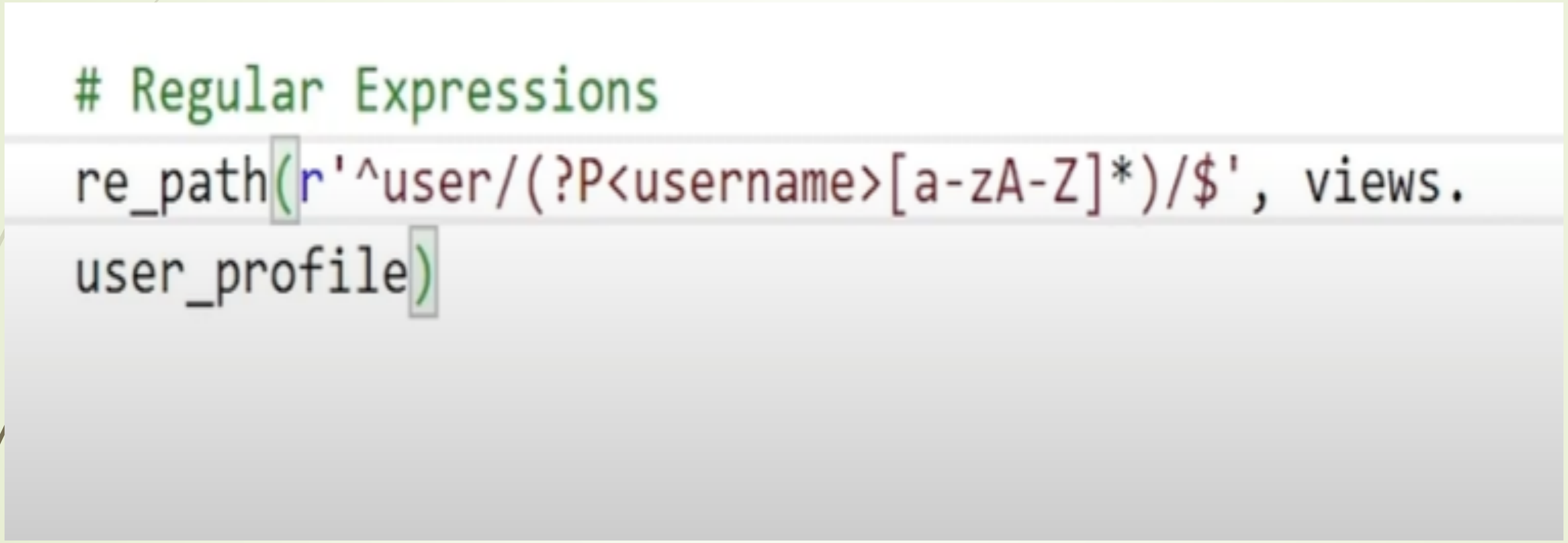
Output:/user/Raj



127.0.0.1:8000/user/Raj/

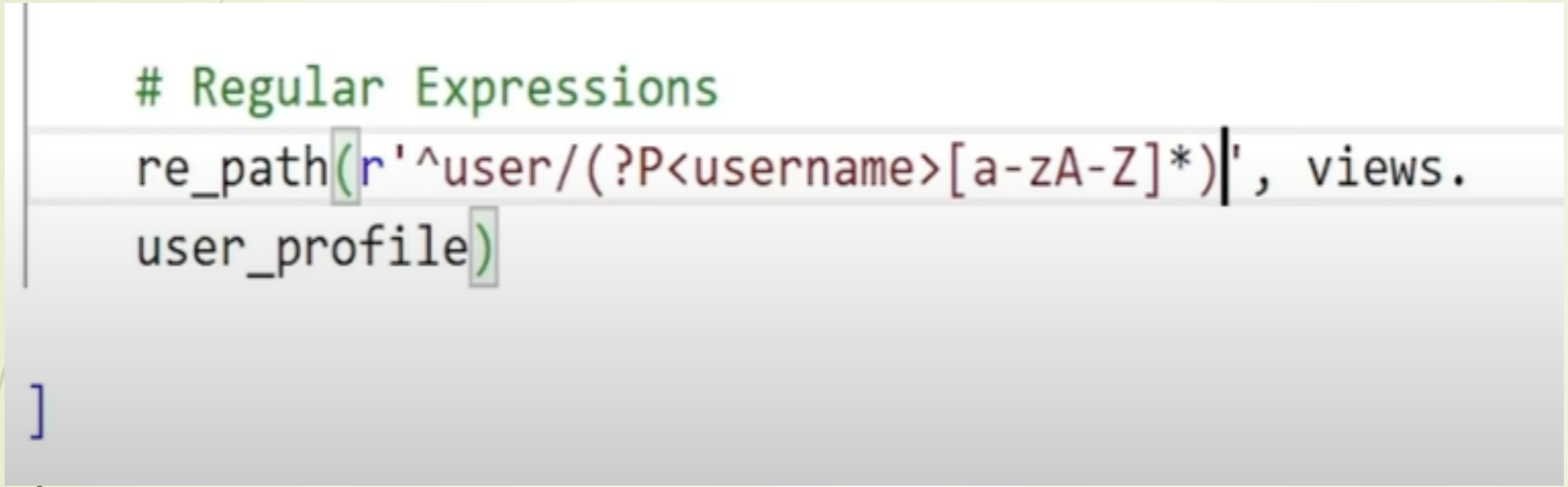
```
User profile: Raj
```

Making the name in the url optional



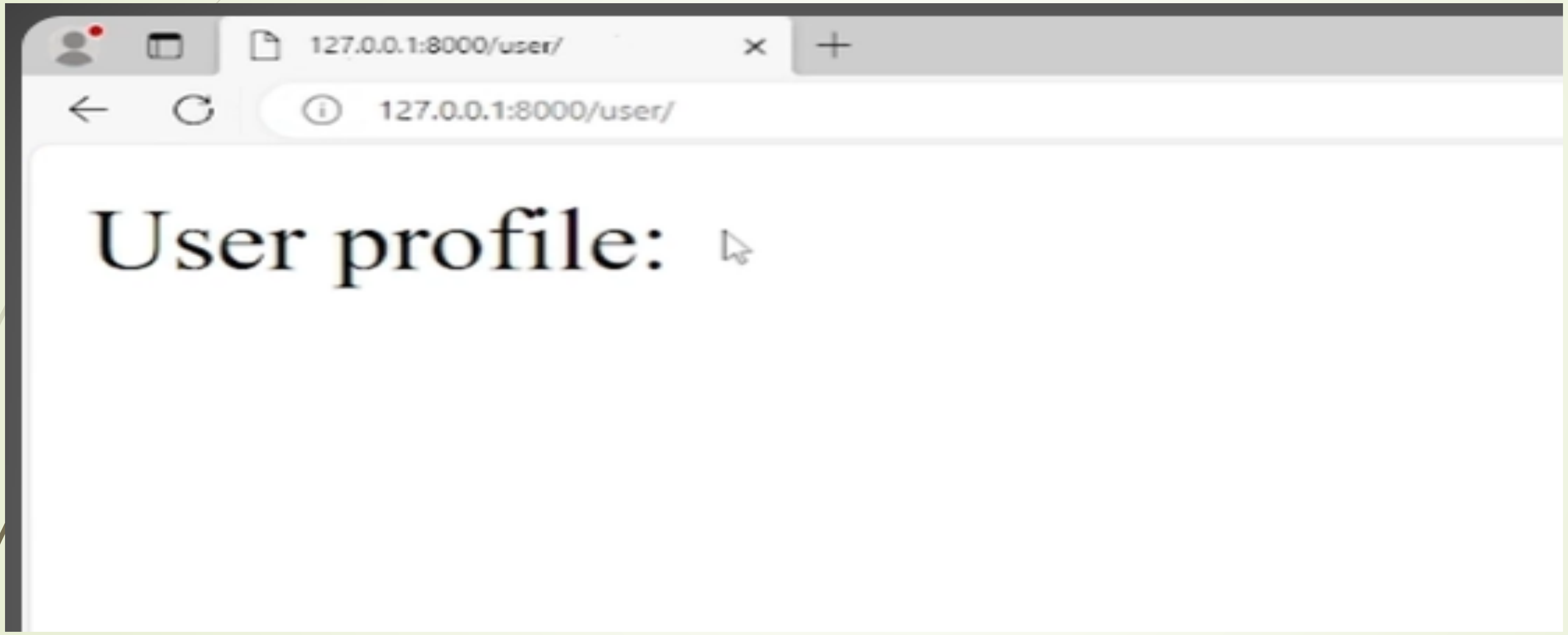
```
# Regular Expressions
re_path(r'^user/(?P<username>[a-zA-Z]*)/$', views.
user_profile)
```

Removing End marker



```
    # Regular Expressions
    re_path(r'^user/(?P<username>[a-zA-Z]*)', views.
    user_profile)

]
```

Output:/user/

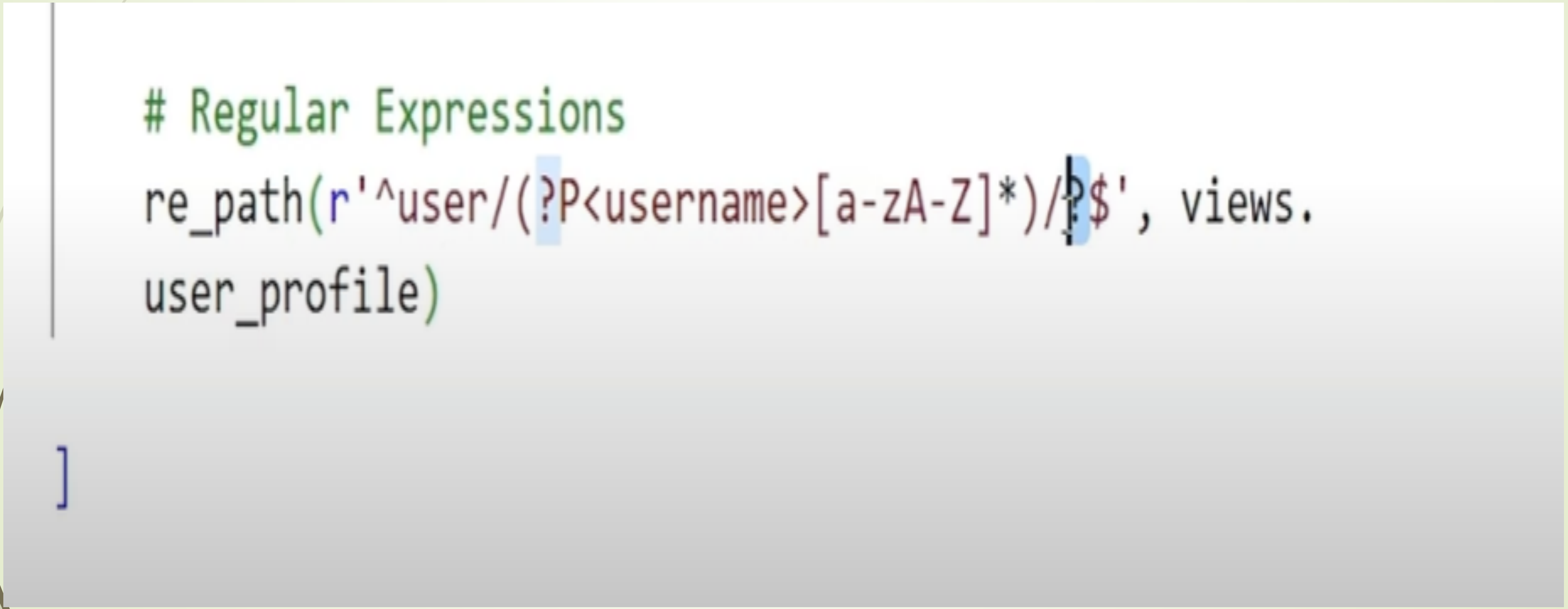


127.0.0.1:8000/user/

127.0.0.1:8000/user/

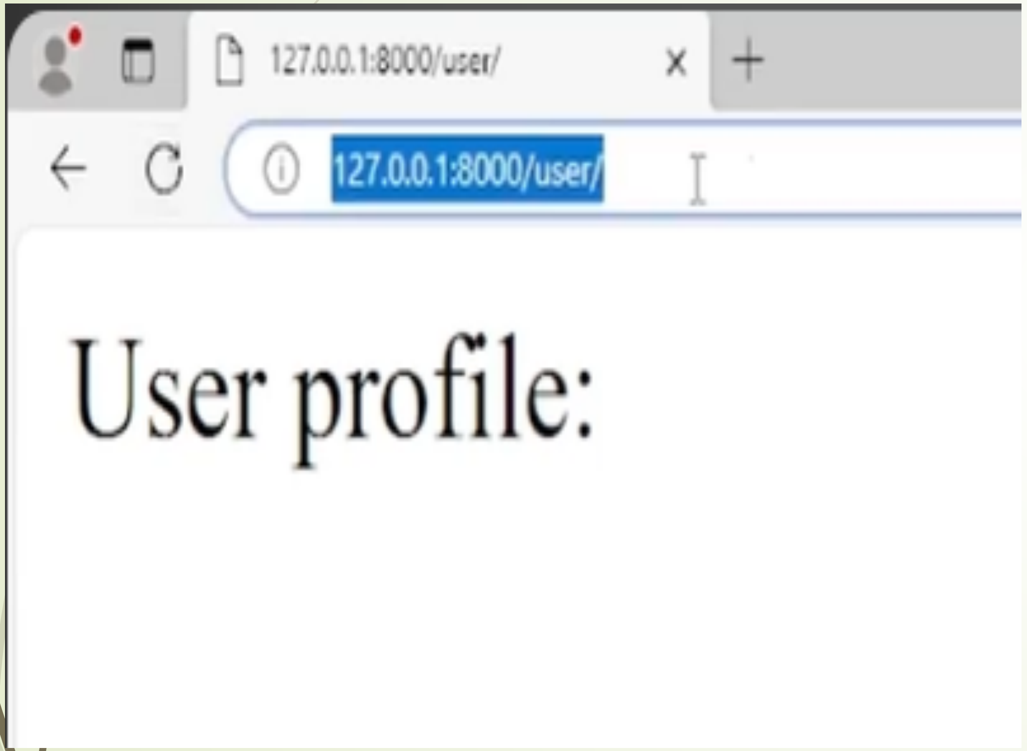
User profile:

Putting ? Making the end marker optional



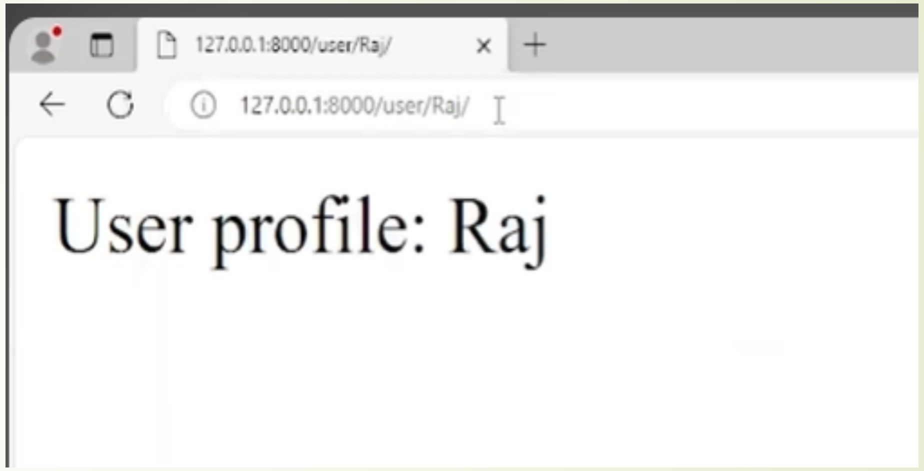
```
# Regular Expressions
re_path(r'^user/(?P<username>[a-zA-Z]*)/?$', views.
user_profile)

]
```

Output :Corresponding to ?



127.0.0.1:8000/user/

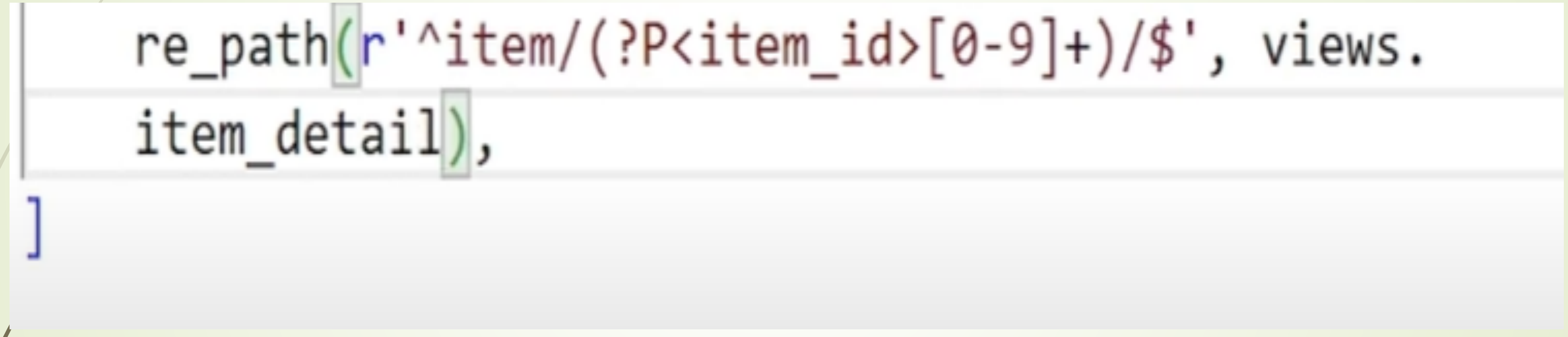
```
User profile:
```



127.0.0.1:8000/user/Raj/

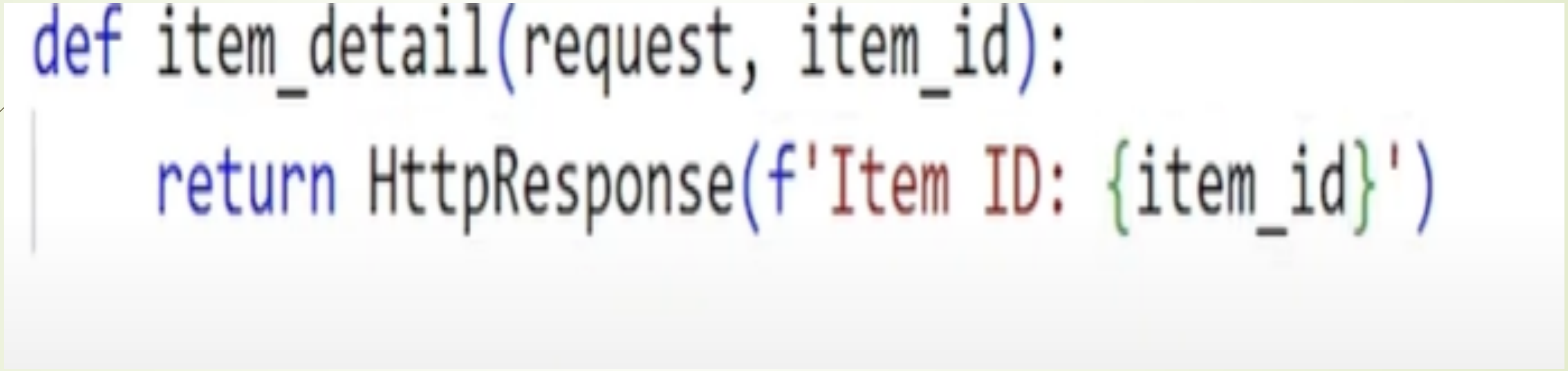
```
User profile: Raj
```

Another path in url.py



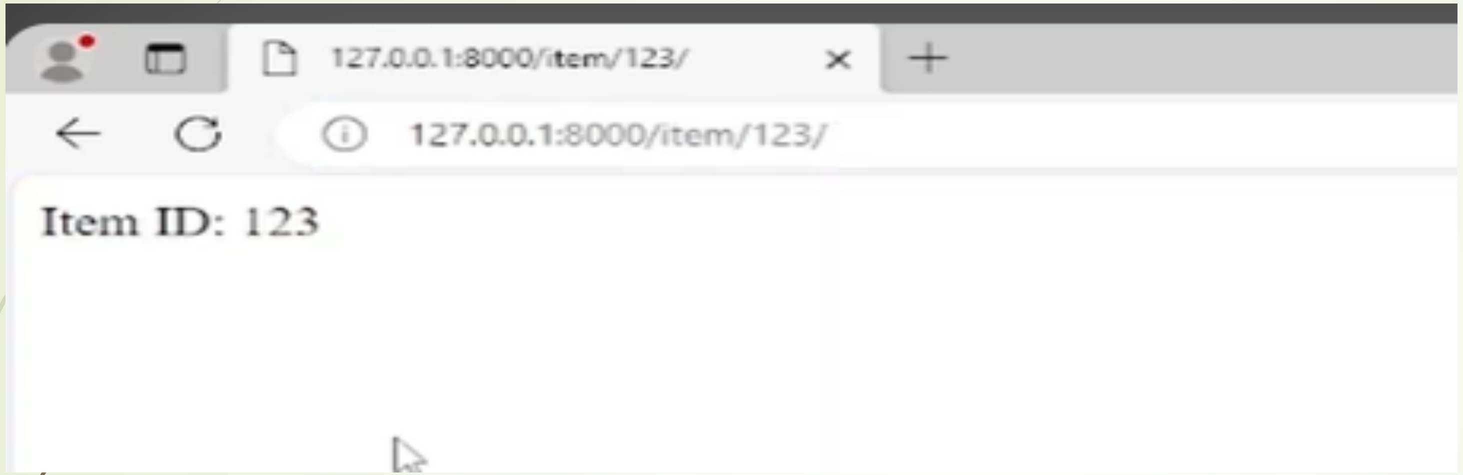
```
    re_path(r'^item/(?P<item_id>[0-9]+)/$', views.
    item_detail),
]
```

Creating Item _detail in view.py



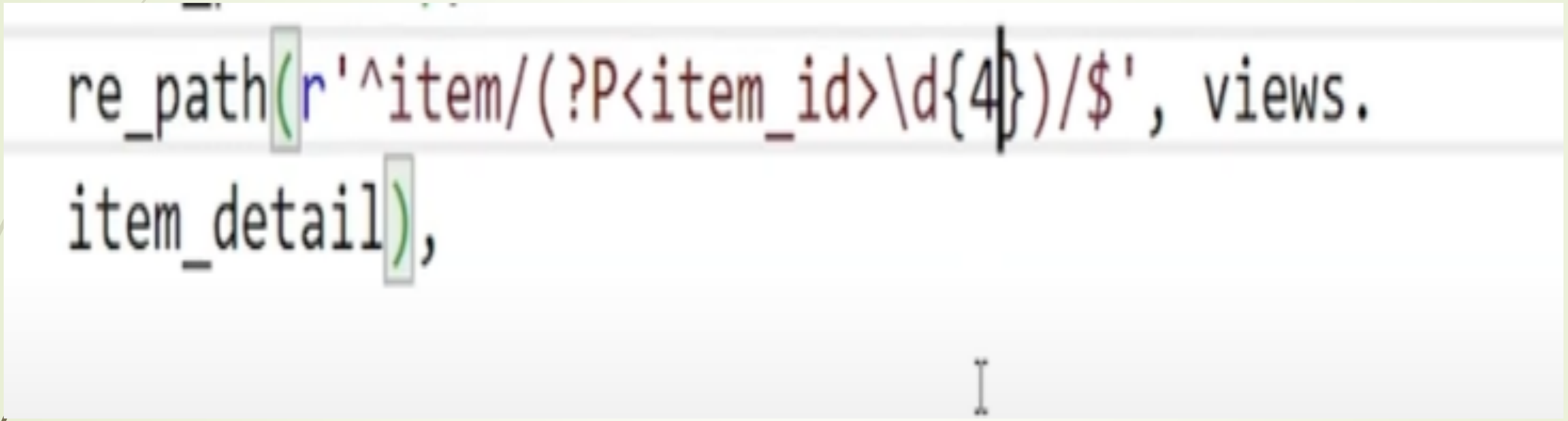
```python
def item_detail(request, item_id):
    return HttpResponse(f'Item ID: {item_id}')
```

OUTPUT

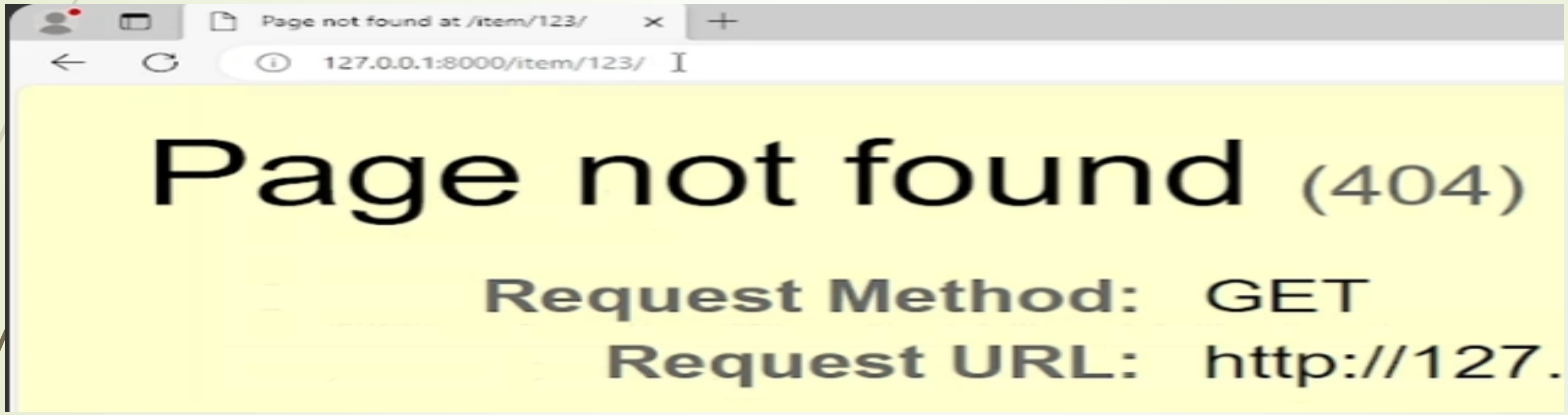


```
Item ID: 123
```

View.py


```
re_path(r'^item/(?P<item_id>\d{4})/$', views.
item_detail),
```

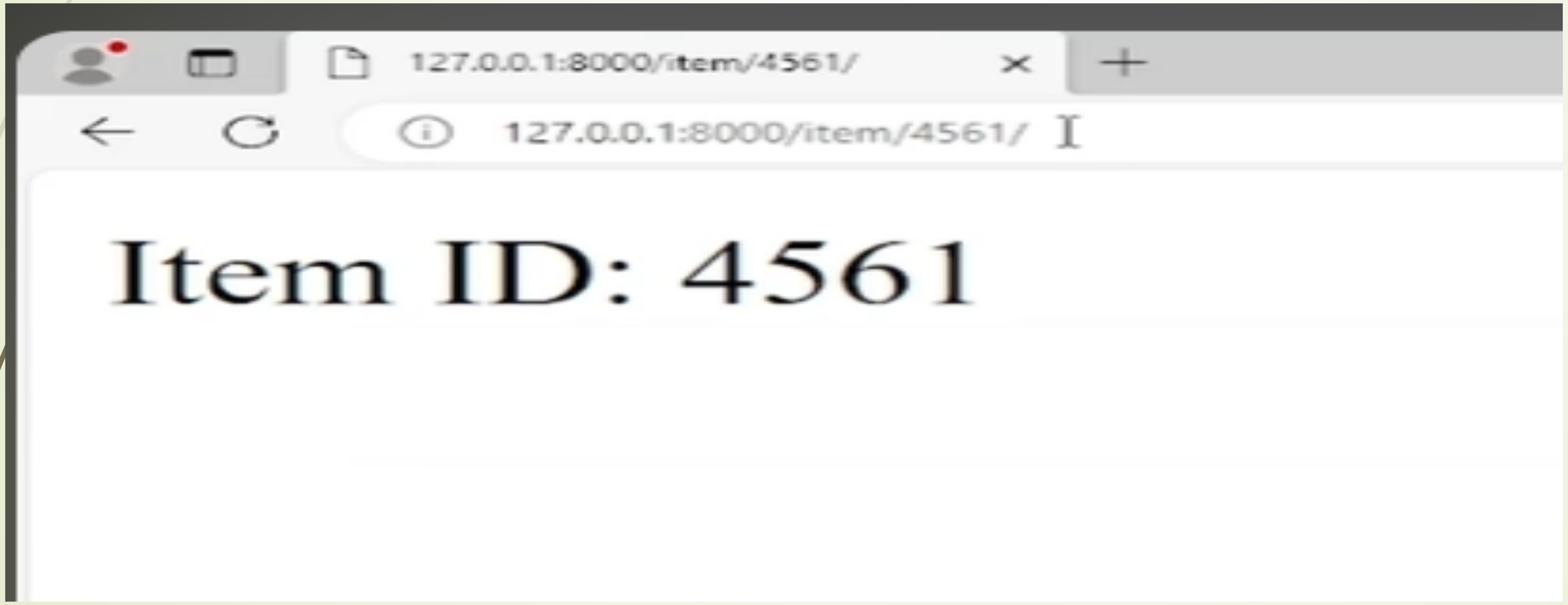
Output 127.0.0.1:8000/item/123/



Page not found (404)

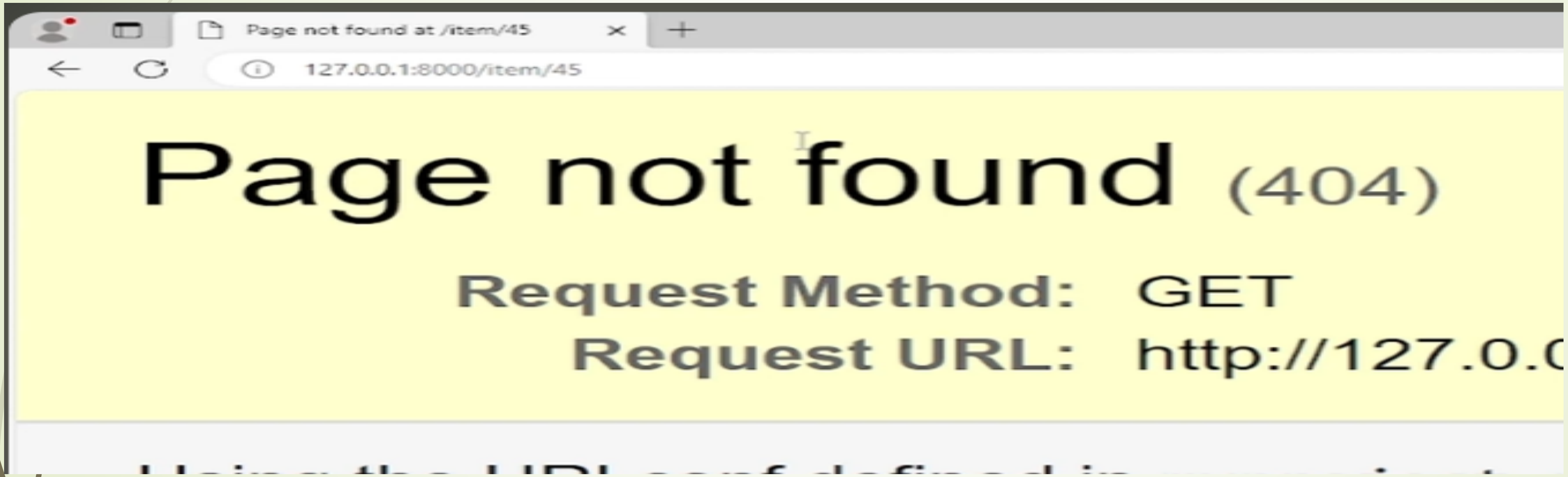
Request Method: GET
Request URL: http://127.

127.0.0.1:8000/item/4561



Item ID: 4561

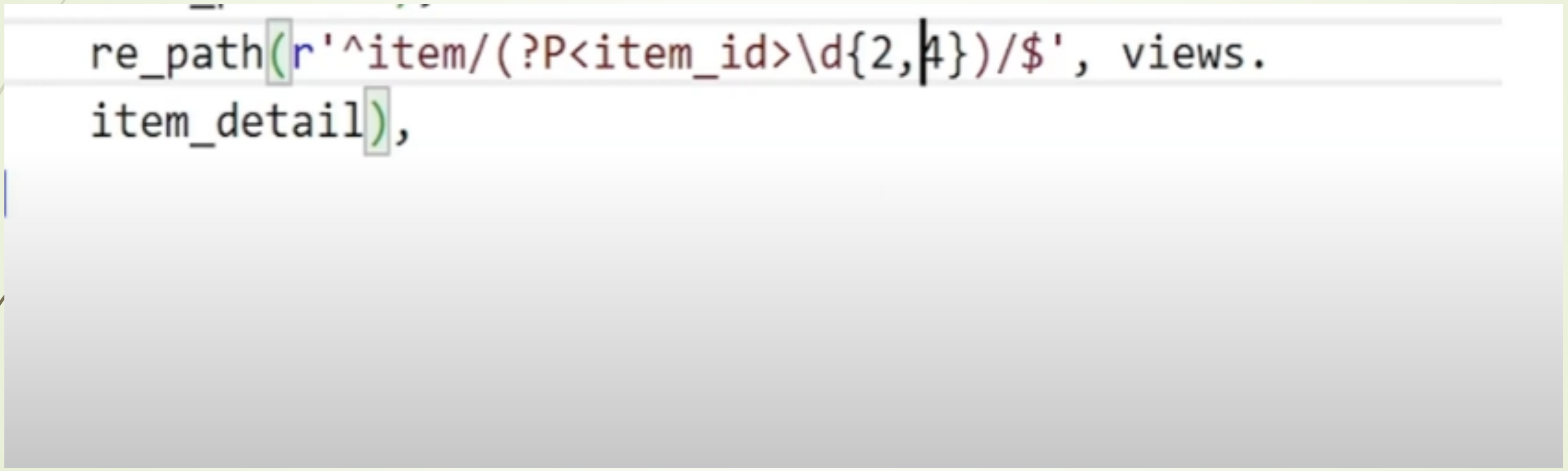
127.0.0.1:8000/item/45



Page not found (404)

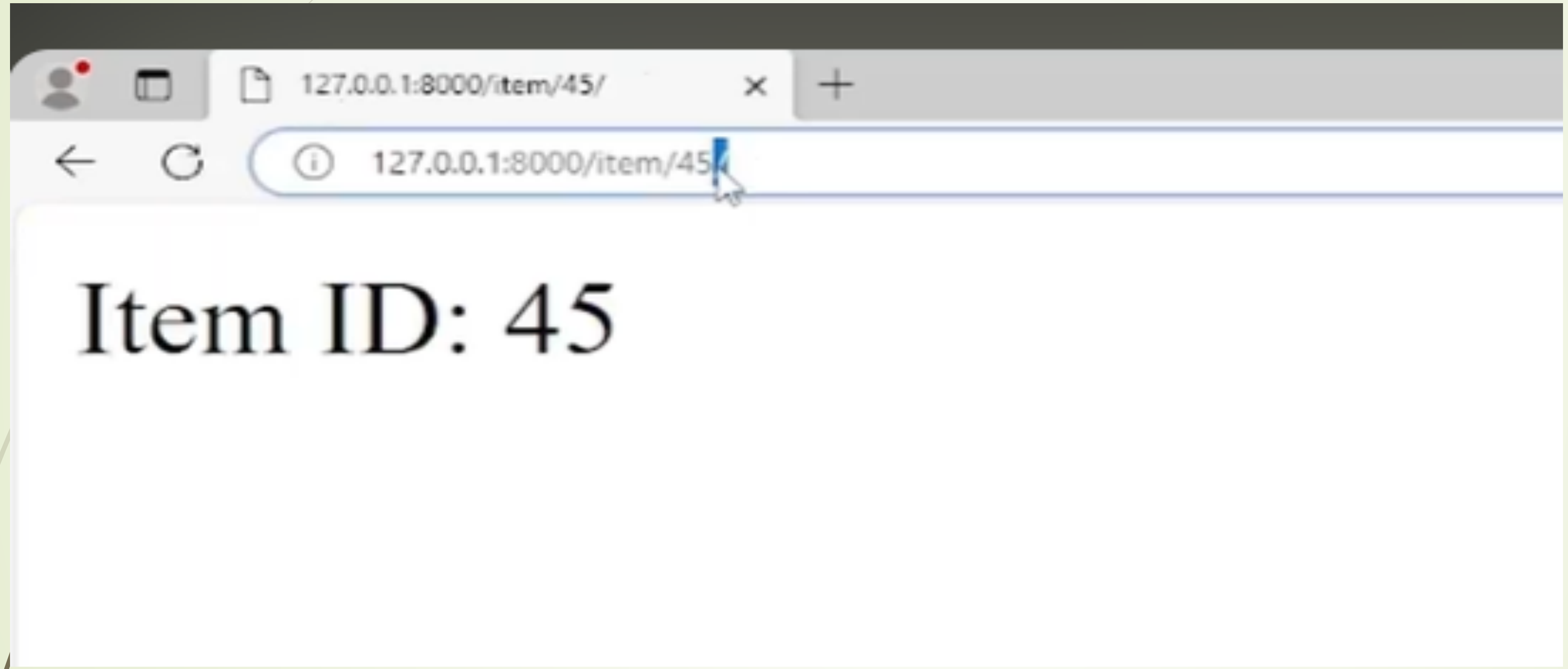
Request Method: GET
Request URL: http://127.0.0.

Setting upper and lower limit



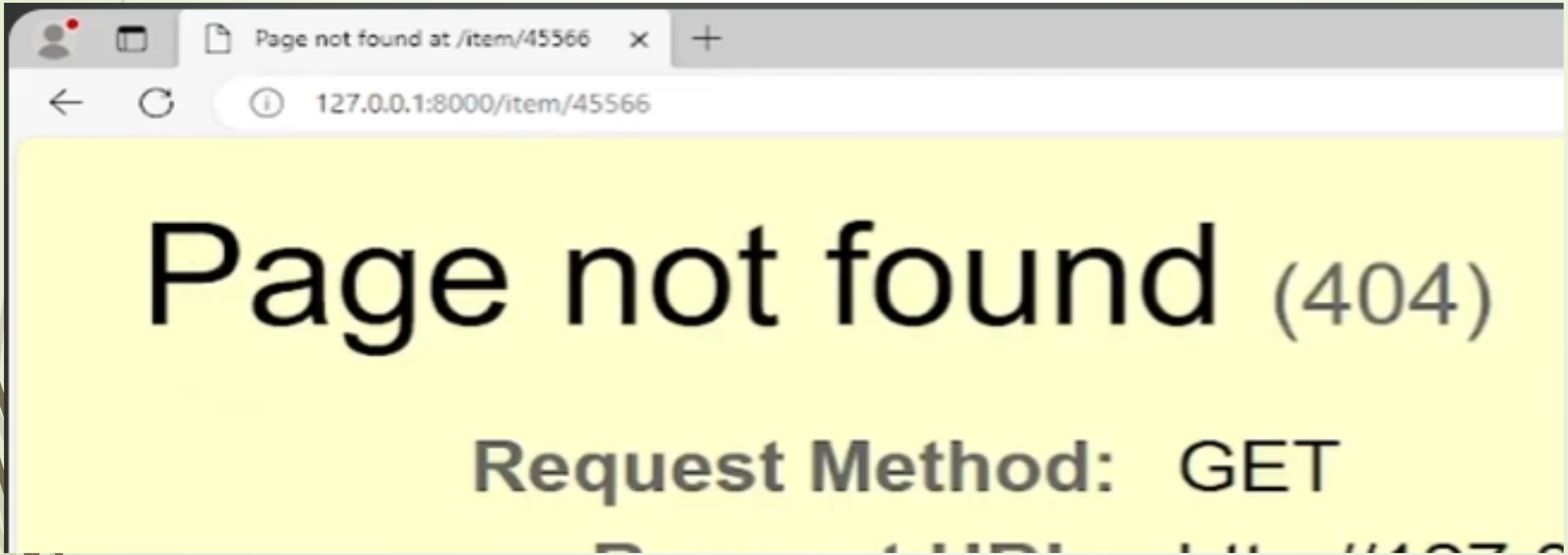
```
re_path(r'^item/(?P<item_id>\d{2,4})/$', views.
item_detail),
```

OUTPUT



```
Item ID: 45
```

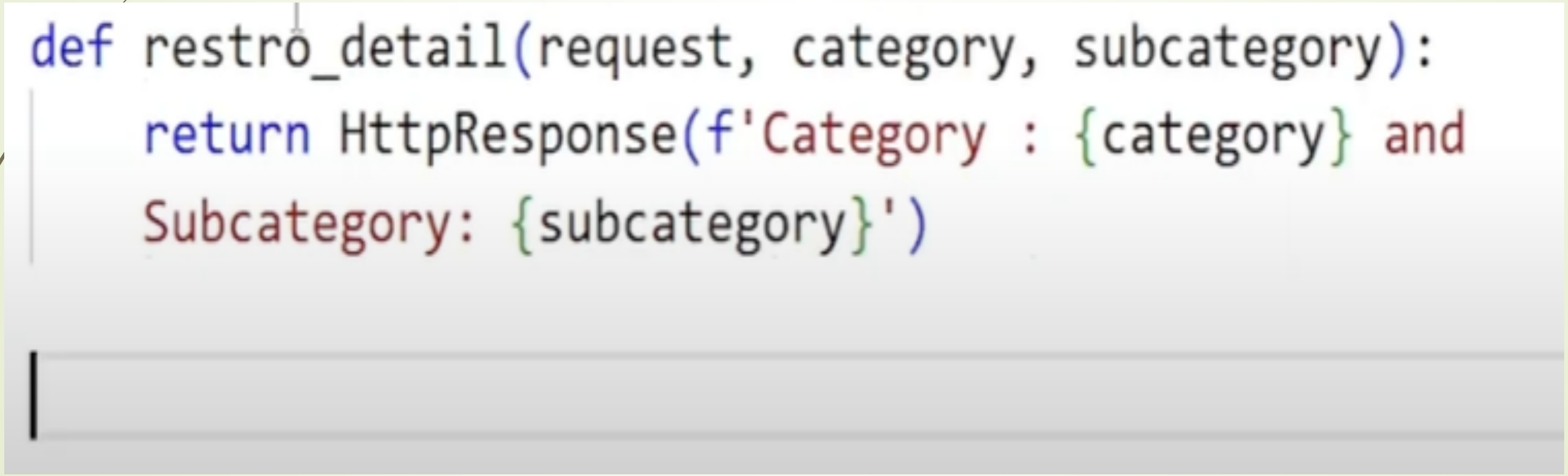
127.0.0:8000/item/45566



Page not found (404)

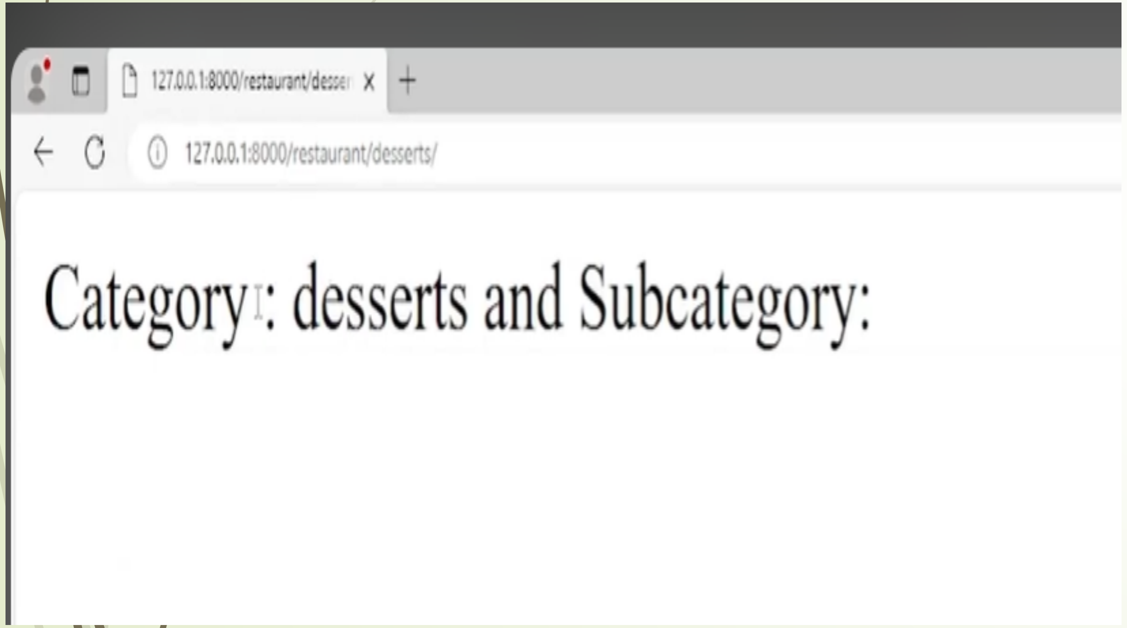
Request Method: GET

Creating a category and sub category



```
def restro_detail(request, category, subcategory):
    return HttpResponse(f'Category : {category} and
    Subcategory: {subcategory}')
```

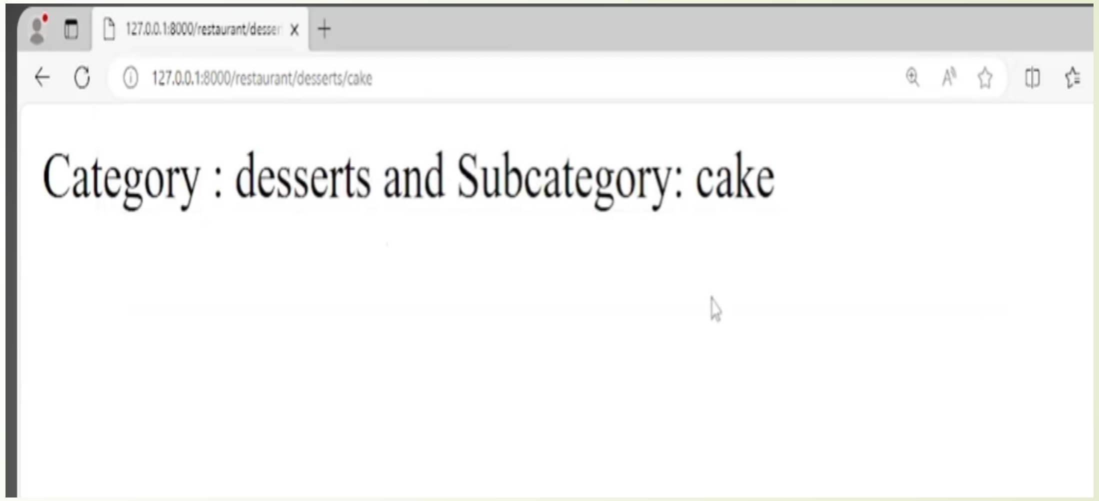
Output



127.0.0.1:8000/restaurant/desser

127.0.0.1:8000/restaurant/desserts/

```
Category : desserts and Subcategory:
```

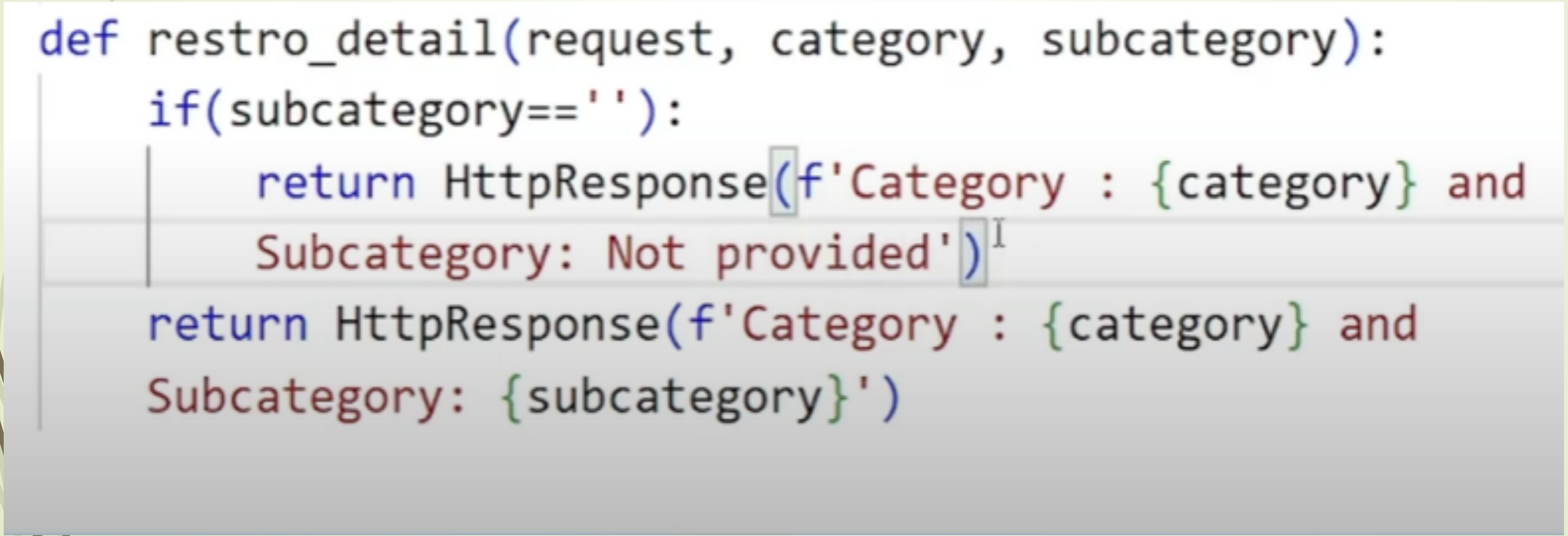


127.0.0.1:8000/restaurant/desser

127.0.0.1:8000/restaurant/desserts/cake

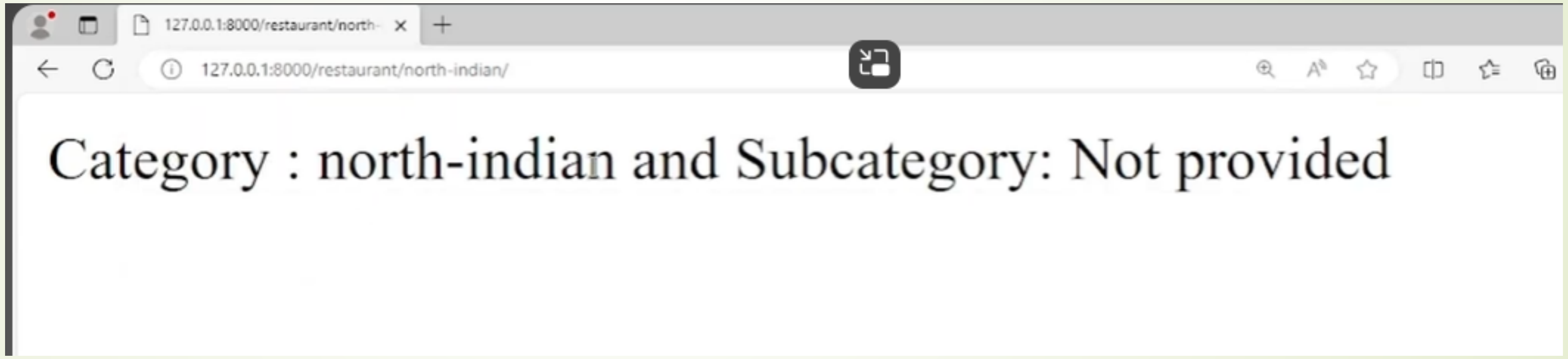
```
Category : desserts and Subcategory: cake
```

In Case subcategories not mentioned

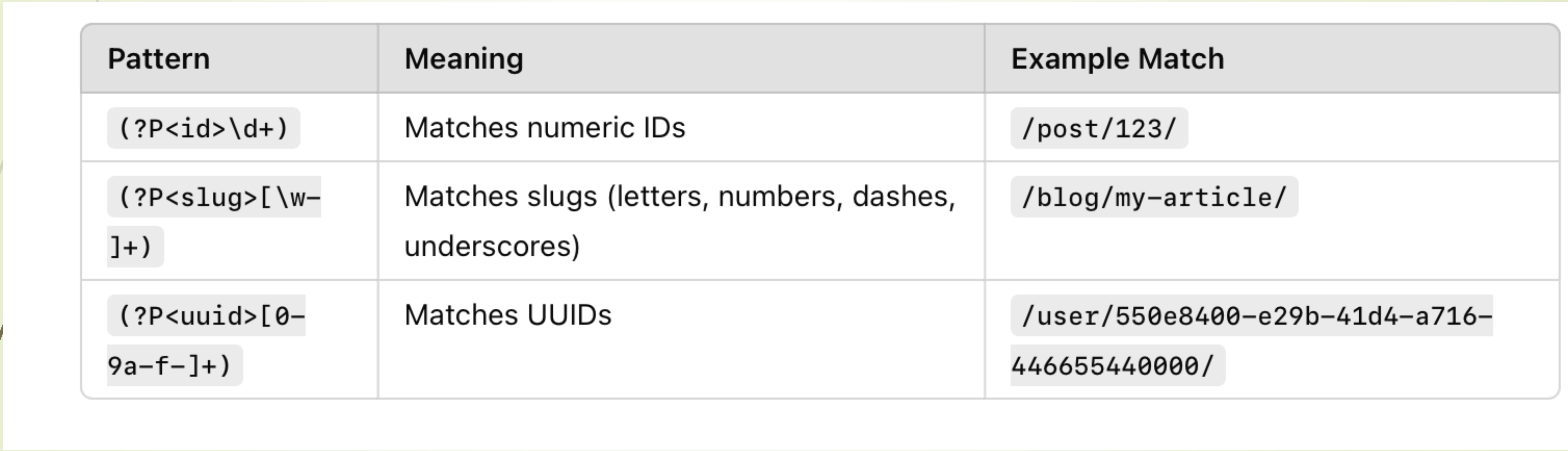


```
def restro_detail(request, category, subcategory):
    if(subcategory==''):
        return HttpResponse(f'Category : {category} and
        Subcategory: Not provided')
    return HttpResponse(f'Category : {category} and
    Subcategory: {subcategory}')
```

Output 127.0.0.1:8000/restaurant/north-indian


127.0.0.1:8000/restaurant/north-
127.0.0.1:8000/restaurant/north-indian/

```
Category : north-indian and Subcategory: Not provided
```



Pattern	Meaning	Example Match
`(?P<id>\d+)`	Matches numeric IDs	`/post/123/`
`(?P<slug>[\w-]+)`	Matches slugs (letters, numbers, dashes, underscores)	`/blog/my-article/`
`(?P<uuid>[0-9a-f-]+)`	Matches UUIDs	`/user/550e8400-e29b-41d4-a716-446655440000/`

URL Pattern	Regex Used
/user/123/	\d+
/blog/hello-world/	[-\w]+
/archive/2025-02-01/	\d{4}-\d{2}-\d{2}
/shop/electronics/laptop/	\w+/\w+
/product/A1B2C3/	[A-Za-z0-9]{6}

When to Use `re_path()`?

- When defining complex patterns (e.g., matching multiple formats in one URL)
- When migrating from older Django versions that use regex-based URLs

```
from django.urls import re_path
from myapp import views
urlpatterns = [ re_path(r'^article/(?P<year>[0-9]
{4})/$', views.article_detail), ]
```

You are building a Django application that handles blog posts.

You need to create a URL pattern using `re_path()` that matches the following conditions:

1. The URL should follow the format: `/blog/yyyy/mm/dd/slug/`
2. `yyyy` should be a 4-digit year (e.g., `2024`).
3. `mm` should be a 2-digit month (01-12).
4. `dd` should be a 2-digit day (01-31).
5. `slug` should be a string containing letters, numbers, dashes, and underscores.

Write the Django `urlpatterns` entry for this.

```
from django.urls import re_path
 from myapp import views
 urlpatterns = [ re_path( r'^blog/(?P<year>\d{4})/(?
P<month>0[1-9]|1[0-2])/(?P<day>0[1-9]|[12][0-9]|3[01])/(?
P<slug>[\w-]+)/$', views.blog_detail ), ]
```

`^blog/` → The URL must start with "blog/"

`(?P<year>\d{4})` → Captures a **4-digit year** (`\d{4}`) as `"year"`

`(?P<month>0[1-9]|1[0-2])` → Captures a **2-digit month** (`01-12`)

`(?P<day>0[1-9]|[12][0-9]|3[01])` → Captures a **valid 2-digit day** (`01-31`)

`(?P<slug>[\w-]+)` → Captures a **slug** containing letters, numbers, dashes, and underscores

`/$` → Ensures the URL ends with `/`

Example Matches:

✅ `/blog/2024/03/15/my-first-post/` → **Valid**
✅ `/blog/1999/12/01/django-url-routing/` → **Valid**
❌ `/blog/2024/13/05/invalid-month/` → **Invalid** (Month must be `01-12`)
❌ `/blog/2024/02/30/fake-day/` → **Invalid** (No `30th Feb`)

Alternative Using `path()`

```
from django.urls import path
 from myapp import views
 urlpatterns = [ path('blog/<int:year>/<int:month>/<int:day>/<slug:slug>/', views.blog_detail), ]
```

Create a Django URL pattern that matches URLs like:

- `/shop/electronics/laptop/`
- `/shop/clothing/shirt/`
-

Views.py

```
from django.urls import re_path
from django.http import HttpResponse
 def shop(request, category, item):
return HttpResponse(f"Category: {category}, Item: {item}")
```

url.py

```
urlpatterns = [ re_path(r'^shop/(?P<category>\w+)/(?P<item>\w+)/$', shop), ]
```