Query Parameters

When you declare parameters in function which are not a part of path parameters, they are automatically considered as query parameters.

```
@app.get('/products/')
def products(id):
    return {f'Product with an id {id}'}
```

Now visit:

```
http://127.0.0.1:8000/products/
```

This will give an error as now we need to pass in a query parameter

Unlike path parameters, you cannot just say/query_parameter, instead to use a query parameter, you need to use a ? sign.

Now instead pass this request:

```
http://127.0.0.1:8000/products/?id=10
```

Passing multiple parameters.

You can also pass multiple parameters to a function.

```
@app.get('/products/')
def products(id,price):
    return {f'Product with an id {id} and price: {price}'}
```

Make a request as follows:

```
http://127.0.0.1:8000/products/?id=10&price=40
```

Providing default values.

```
@app.get('/products/')
def products(id=10,price=100):
    return {f'Product with an id {id} and price: {price}'}
```

Now visit:

```
http://127.0.0.1:8000/products/
```

Now default values will be placed in place of query parameters even if they are not passed in the URL.

Providing data types for query parameters.

```
@app.get('/products/')
def products(id:int=10,price:int=100):
    return {f'Product with an id {id} and price: {price}'}
```

Multiple path and query parameters

```
@app.get('/profile/{userid}/comments/')
def profile(userid:int,commentid:int):
    return {f'Profile page for user with userid {userid} and comment with id {commentid}'}
```

Required query parameters

When you pass a default value to parameter, it becomes optional.

But if you don't want to add a specific value but make it optional, in that case set the default as None.

As follows:

But when you want to make a query parameter required, you can just not declare any default value: