Body - Fields

The same way you can declare additional validation and metadata in *path operation function* parameters with `Query`, `Path` and `Body`, you can declare validation and metadata inside of Pydantic models using Pydantic's `Field`.

Import `Field`

First, you have to import it:

=== "Python 3.6 and above"

~~~
```Python hl_lines="4"
{!> ../../../docs_src/body_fields/tutorial001.py!}
```
~~~

=== "Python 3.10 and above"

~~~
```Python hl_lines="2"
{!> ../../../docs_src/body_fields/tutorial001_py310.py!}
```
~~~

!!! warning Notice that `Field` is imported directly from `pydantic`, not from `fastapi` as are all the rest (`Query`, `Path`, `Body`, etc).

Declare model attributes

You can then use `Field` with model attributes:

=== "Python 3.6 and above"

~~~
```Python hl_lines="11-14"
{!> ../../../docs_src/body_fields/tutorial001.py!}
```
~~~

=== "Python 3.10 and above"

~~~
```Python hl_lines="9-12"
{!> ../../../docs_src/body_fields/tutorial001_py310.py!}
```
~~~

`Field` works the same way as `Query`, `Path` and `Body`, it has all the same parameters, etc.

!!! note "Technical Details" Actually, `Query`, `Path` and others you'll see next create objects of subclasses of a common `Param` class, which is itself a subclass of Pydantic's `FieldInfo` class.

~~~
And Pydantic's `Field` returns an instance of `FieldInfo` as well.

`Body` also returns objects of a subclass of `FieldInfo` directly. And there are others you
~~~

```
Remember that when you import `Query`, `Path`, and others from `fastapi`, those are actually
```

!!! tip Notice how each model's attribute with a type, default value and `Field` has the same structure as a *path operation function's* parameter, with `Field` instead of `Path`, `Query` and `Body`.

Add extra information

You can declare extra information in `Field`, `Query`, `Body`, etc. And it will be included in the generated JSON Schema.

You will learn more about adding extra information later in the docs, when learning to declare examples.

Recap

You can use Pydantic's `Field` to declare extra validations and metadata for model attributes.

You can also use the extra keyword arguments to pass additional JSON Schema metadata.