# Header Parameters

You can define Header parameters the same way you define `Query`, `Path` and `Cookie` parameters.

## Import `Header`

First import `Header`:

=== "Python 3.6 and above"

```Python hl_lines="3"
{!> ../../../docs_src/header_params/tutorial001.py!}
```

=== "Python 3.10 and above"

```Python hl_lines="1"
{!> ../../../docs_src/header_params/tutorial001_py310.py!}
```

## Declare `Header` parameters

Then declare the header parameters using the same structure as with `Path`, `Query` and `Cookie`.

The first value is the default value, you can pass all the extra validation or annotation parameters:

=== "Python 3.6 and above"

```Python hl_lines="9"
{!> ../../../docs_src/header_params/tutorial001.py!}
```

=== "Python 3.10 and above"

```Python hl_lines="7"
{!> ../../../docs_src/header_params/tutorial001_py310.py!}
```

!!! note "Technical Details" `Header` is a "sister" class of `Path`, `Query` and `Cookie`. It also inherits from the same common `Param` class.

`But remember that when you import `Query`, `Path`, `Header`, and others from `fastapi`, thos`

!!! info To declare headers, you need to use `Header`, because otherwise the parameters would be interpreted as query parameters.

## Automatic conversion

`Header` has a little extra functionality on top of what `Path`, `Query` and `Cookie` provide.

Most of the standard headers are separated by a "hyphen" character, also known as the "minus symbol" (`-`).

But a variable like `user-agent` is invalid in Python.

So, by default, `Header` will convert the parameter names characters from underscore (`_`) to hyphen (`-`) to extract and document the headers.

Also, HTTP headers are case-insensitive, so, you can declare them with standard Python style (also known as "snake_case").

So, you can use `user_agent` as you normally would in Python code, instead of needing to capitalize the first letters as `User_Agent` or something similar.

If for some reason you need to disable automatic conversion of underscores to hyphens, set the parameter `convert_underscores` of `Header` to `False`:

=== "Python 3.6 and above"

```Python hl_lines="10"
{!> ../../../docs_src/header_params/tutorial002.py!}
```

=== "Python 3.10 and above"

```Python hl_lines="8"
{!> ../../../docs_src/header_params/tutorial002_py310.py!}
```

!!! warning Before setting `convert_underscores` to `False`, bear in mind that some HTTP proxies and servers disallow the usage of headers with underscores.

## Duplicate headers

It is possible to receive duplicate headers. That means, the same header with multiple values.

You can define those cases using a list in the type declaration.

You will receive all the values from the duplicate header as a Python `list`.

For example, to declare a header of `X-Token` that can appear more than once, you can write:

=== "Python 3.6 and above"

```Python hl_lines="9"
{!> ../../../docs_src/header_params/tutorial003.py!}
```

=== "Python 3.9 and above"

```Python hl_lines="9"
{!> ../../../docs_src/header_params/tutorial003_py39.py!}
```

=== "Python 3.10 and above"

```Python hl_lines="7"
{!> ../../../docs_src/header_params/tutorial003_py310.py!}
```

If you communicate with that *path operation* sending two HTTP headers like:

```
X-Token: foo
X-Token: bar
```

The response would be like:

```
{
    "X-Token values": [
        "bar",
        "foo"
    ]
}
```

## Recap

Declare headers with `Header`, using the same common pattern as `Query`, `Path` and `Cookie`.

And don't worry about underscores in your variables, **FastAPI** will take care of converting them.