# 경로 매개변수와 숫자 검증

`Query`를 사용하여 쿼리 매개변수에 더 많은 검증과 메타데이터를 선언하는 방법과 동일하게 `Path`를 사용하여 경로 매개변수에 검증과 메타데이터를 같은 타입으로 선언할 수 있습니다.

## 경로 임포트

먼저 `fastapi`에서 `Path`를 임포트합니다:

```
{!../../../docs_src/path_params_numeric_validations/tutorial001.py!}
```

## 메타데이터 선언

`Query`에 동일한 매개변수를 선언할 수 있습니다.

예를 들어, `title` 메타데이터 값을 경로 매개변수 `item_id`에 선언하려면 다음과 같이 입력할 수 있습니다:

```
{!../../../docs_src/path_params_numeric_validations/tutorial001.py!}
```

!!! note "참고" 경로 매개변수는 경로의 일부여야 하므로 언제나 필수적입니다.

즉, `...`로 선언해서 필수임을 나타내는게 좋습니다.

그럼에도 `None`으로 선언하거나 기본값을 지정할지라도 아무 영향을 끼치지 않으며 언제나 필수입니다.

## 필요한 경우 매개변수 정렬하기

`str` 형인 쿼리 매개변수 `q`를 필수로 선언하고 싶다고 해봅시다.

해당 매개변수에 대해 아무런 선언을 할 필요가 없으므로 `Query`를 정말로 써야할 필요는 없습니다.

하지만 `item_id` 경로 매개변수는 여전히 `Path`를 사용해야 합니다.

파이썬은 "기본값"이 없는 값 앞에 "기본값"이 있는 값을 입력하면 불평합니다.

그러나 매개변수들을 재정렬함으로써 기본값(쿼리 매개변수 `q`)이 없는 값을 처음 부분에 위치 할 수 있습니다.

**FastAPI**에서는 중요하지 않습니다. 이름, 타입 그리고 선언구(`Query`, `Path` 등)로 매개변수를 감지하며 순서는 신경 쓰지 않습니다.

따라서 함수를 다음과 같이 선언 할 수 있습니다:

```
{!../../../docs_src/path_params_numeric_validations/tutorial002.py!}
```

## 필요한 경우 매개변수 정렬하기, 트릭

`Query`나 아무런 기본값으로도 `q` 경로 매개변수를 선언하고 싶지 않지만 `Path`를 사용하여 경로 매개변수를 `item_id` 다른 순서로 선언하고 싶다면, 파이썬은 이를 위한 작고 특별한 문법이 있습니다.

`*`를 함수의 첫 번째 매개변수로 전달하세요.

파이썬은 `*` 으로 아무런 행동도 하지 않지만, 따르는 매개변수들은 `kwargs` 로도 알려진 키워드 인자(키-값 쌍)여야 함을 인지합니다. 기본값을 가지고 있지 않더라도 그렇습니다.

```
{!../../../docs_src/path_params_numeric_validations/tutorial003.py!}
```

## 숫자 검증: 크거나 같음

`Query` 와 `Path` (나중에 볼 다른 것들도)를 사용하여 문자열 뿐만 아니라 숫자의 제약을 선언할 수 있습니다.

여기서 `ge=1` 인 경우, `item_id` 는 `1` 보다 "크거나( `g` reater) 같은( `e` qual)" 정수형 숫자여야 합니다.

```
{!../../../docs_src/path_params_numeric_validations/tutorial004.py!}
```

## 숫자 검증: 크거나 같음 및 작거나 같음

동일하게 적용됩니다:

- `gt` : 크거나( `g` reater `t` han)
- `le` : 작거나 같은( `l` ess than or `e` qual)

```
{!../../../docs_src/path_params_numeric_validations/tutorial005.py!}
```

## 숫자 검증: 부동소수, 크거나 및 작거나

숫자 검증은 `float` 값에도 동작합니다.

여기에서 `ge` 뿐만 아니라 `gt` 를 선언 할 수있는 것이 중요해집니다. 예를 들어 필요한 경우, 값이 `1` 보다 작더라도 반드시 `0` 보다 커야합니다.

즉, `0.5` 는 유효한 값입니다. 그러나 `0.0` 또는 `0` 은 그렇지 않습니다.

`lt` 역시 마찬가지입니다.

```
{!../../../docs_src/path_params_numeric_validations/tutorial006.py!}
```

## 요약

`Query` , `Path` (아직 보지 못한 다른 것들도)를 사용하면 [쿼리 매개변수와 문자열 검증]{.internal-link target=_blank} 에서와 마찬가지로 메타데이터와 문자열 검증을 선언할 수 있습니다.

그리고 숫자 검증 또한 선언할 수 있습니다:

- `gt` : 크거나( `g` reater `t` han)
- `ge` : 크거나 같은( `g` reater than or `e` qual)
- `lt` : 작거나( `l` ess `t` han)
- `le` : 작거나 같은( `l` ess than or `e` qual)

!!! info "정보" `Query` , `Path` , 그리고 나중에게 보게될 것들은 (여러분이 사용할 필요가 없는) 공통 `Param` 클래스의 서브 클래스입니다.

그리고 이들 모두는 여태까지 본 추가 검증과 메타데이터의 동일한 모든 매개변수를 공유합니다.

!!! note "기술 세부사항" `fastapi`에서 `Query`, `Path` 등을 임포트 할 때, 이것들은 실제로 함수입니다.

호출되면 동일한 이름의 클래스의 인스턴스를 반환합니다.

즉, 함수인 `Query`를 임포트한 겁니다. 그리고 호출하면 `Query`라는 이름을 가진 클래스의 인스턴스를 반환합니다.

편집기에서 타입에 대한 오류를 표시하지 않도록 하기 위해 (클래스를 직접 사용하는 대신) 이러한 함수들이 있습니다.

이렇게 하면 오류를 무시하기 위한 사용자 설정을 추가하지 않고도 일반 편집기와 코딩 도구를 사용할 수 있습니다.