Python Veri Tiplerine Giriş

Python isteğe bağlı olarak "tip belirteçlerini" destekler.

"Tip belirteçleri" bir değişkenin tipinin belirtilmesine olanak sağlayan özel bir sözdizimidir.

Değişkenlerin tiplerini belirterek editör ve araçlardan daha fazla destek alabilirsiniz.

Bu pythonda tip belirteçleri için **hızlı bir başlangıç / bilgi tazeleme** rehberidir . Bu rehber **FastAPI** kullanmak için gereken minimum konuyu kapsar ki bu da çok az bir miktardır.

FastAPI' nin tamamı bu tür tip belirteçleri ile donatılmıştır ve birçok avantaj sağlamaktadır.

FastAPI kullanmayacak olsanız bile tür belirteçleri hakkında bilgi edinmenizde fayda var.

!!! not Python uzmanıysanız ve tip belirteçleri ilgili her şeyi zaten biliyorsanız, sonraki bölüme geçin.

Motivasyon

Basit bir örnek ile başlayalım:

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

Programın çıktısı:

```
John Doe
```

Fonksiyon sırayla şunları yapar:

- `first_name` ve `last_name` değerlerini alır.
- `title()` ile değişkenlerin ilk karakterlerini büyütür.
- Değişkenleri aralarında bir boşlukla beraber Birleştirir.

```
Python hl_lines="2" {!../../../docs_src/python_types/tutorial001.py!}
```

Düzenle

Bu çok basit bir program.

Ama şimdi sıfırdan yazdığınızı hayal edin.

Bir noktada fonksiyonun tanımına başlayacaktınız, parametreleri hazır hale getirdiniz...

Ama sonra "ilk harfi büyük harfe dönüştüren yöntemi" çağırmanız gerekir.

`upper` mıydı ? Yoksa `uppercase`' mi? `first_uppercase`? `capitalize`?

Ardından, programcıların en iyi arkadaşı olan otomatik tamamlama ile denediniz.

‘first_name’, ardından bir nokta (‘.’) yazıp otomatik tamamlamayı tetiklemek için ‘Ctrl+Space’ tuşlarına bastınız.

Ancak, ne yazık ki, yararlı hiçbir şey elde edemediniz:

Tipleri ekle

Önceki sürümden sadece bir satırı değiştirelim.

Tam olarak bu parçayı, işlevin parametrelerini değiştireceğiz:

```
first_name, last_name
```

ve bu hale getireceğiz:

```
first_name: str, last_name: str
```

Bu kadar.

İşte bunlar “tip belirteçleri”:

```
Python hl_lines="1" {!../../../docs_src/python_types/tutorial002.py!}
```

Bu, aşağıdaki gibi varsayılan değerleri bildirmekle aynı şey değildir:

```
first_name="john", last_name="doe"
```

Bu tamamen farklı birşey

İki nokta üst üste (`:`) kullanıyoruz , eşittir (`=`) değil.

Normalde tip belirteçleri eklemek, kod üzerinde olacakları değiştirmez.

Şimdi programı sıfırdan birdaha yazdığınızı hayal edin.

Aynı noktada, `Ctrl+Space` ile otomatik tamamlamayı tetiklediniz ve şunu görüyorsunuz:

Aradığınızı bulana kadar seçenekleri kaydırabilirsiniz:

Daha fazla motivasyon

Bu fonksiyon, zaten tür belirteçlerine sahip:

```
Python hl_lines="1" {!../../../docs_src/python_types/tutorial003.py!}
```

Editör değişkenlerin tiplerini bildiğinden, yalnızca otomatik tamamlama değil, hata kontrolleri de sağlar:

Artık `age` değişkenini `str(age)` olarak kullanmanız gerektiğini biliyorsunuz:

```
Python hl_lines="2" {!../../../docs_src/python_types/tutorial004.py!}
```

Tip bildirme

Az önce tip belirteçlerinin en çok kullanıldığı yeri gördünüz.

FastAPIile çalışırken tip belirteçlerini en çok kullanacağımız yer yine fonksiyonlardır.

Basit tipler

Yalnızca `str` değil, tüm standart Python tiplerinin bildirebilirsiniz.

Örneğin şunları kullanabilirsiniz:

- `int`
- `float`
- `bool`
- `bytes`

```
Python hl_lines="1" {!../../../docs_src/python_types/tutorial005.py!}
```

Tip parametreleri ile Generic tipler

"dict", "list", "set" ve "tuple" gibi diğer değerleri içerebilen bazı veri yapıları vardır. Ve dahili değerlerinin de tip belirtecleri olabilir.

Bu tipleri ve dahili tpileri bildirmek için standart Python modülünü "typing" kullanabilirsiniz.

Bu tür tip belirteçlerini desteklemek için özel olarak mevcuttur.

`List` Örneğin `str` değerlerden oluşan bir `list` tanımlayalım.

From `typing`, import `List` (büyük harf olan `L` ile):

```
Python hl_lines="1" {!../../../docs_src/python_types/tutorial006.py!}
```

Değişkenin tipini yine iki nokta üstüste (`:`) ile belirleyin.

tip olarak `List` kullanın.

Liste, bazı dahili tipleri içeren bir tür olduğundan, bunları köşeli parantez içine alırsınız:

```
Python hl_lines="4" {!../../../docs_src/python_types/tutorial006.py!}
```

!!! ipucu Köşeli parantez içindeki bu dahili tiplere "tip parametreleri" denir.

```
Bu durumda `str`, `List`e iletilen tür parametresidir.
```

Bunun anlamı şudur: "`items` değişkeni bir `list`tir ve bu listedeki öğelerin her biri bir `str`dir".

Bunu yaparak, düzenleyicinizin listedeki öğeleri işlerken bile destek sağlamasını sağlayabilirsiniz:

Tip belirteçleri olmadan, bunu başarmak neredeyse imkansızdır.

`item` değişkeninin `items` listesindeki öğelerden biri olduğuna dikkat edin.

Ve yine, editör bunun bir `str` olduğunu biliyor ve bunun için destek sağlıyor.

`Tuple` ve `Set` `Tuple` ve `set`lerin tiplerini bildirmek için de aynısını yapıyoruz:

```
Python hl_lines="1  4" {!../../../docs_src/python_types/tutorial007.py!}
```

Bu şu anlama geliyor:

- `items_t` değişkeni sırasıyla `int`, `int`, ve `str` tiplerinden oluşan bir `tuple` türündedir .
- `items_s` ise her öğesi `bytes` türünde olan bir `set` örneğidir.

`Dict` Bir `dict` tanımlamak için virgülle ayrılmış iki parametre verebilirsiniz.

İlk tip parametresi `dict` değerinin `key` değeri içindir.

İkinci parametre ise `dict` değerinin `value` değeri içindir:

```
Python hl_lines="1  4" {!../../../docs_src/python_types/tutorial008.py!}
```

Bu şu anlama gelir:

- `prices` değişkeni `dict` tipindedir:
  - `dict` değişkeninin `key` değeri `str` tipindedir (herbir item'ın "name" değeri).
  - `dict` değişkeninin `value` değeri `float` tipindedir (lherbir item'ın "price" değeri).

`Optional` `Optional` bir değişkenin `str`gibi bir tipi olabileceğini ama isteğe bağlı olarak tipinin `None` olabileceğini belirtir:

```
Python hl_lines="1  4" {!../../../docs_src/python_types/tutorial009.py!}
```

`str` yerine `Optional[str]` kullanmak editorün bu değerin her zaman `str` tipinde değil bazen `None` tipinde de olabileceğini belirtir ve hataları tespit etmemizde yardımcı olur.

Generic tipler Köşeli parantez içinde tip parametreleri alan bu türler, örneğin:

- `List`
- `Tuple`
- `Set`
- `Dict`
- `Optional`
- ...and others.

Generic types yada **Generics** olarak adlandırılır.

Tip olarak Sınıflar

Bir değişkenin tipini bir sınıf ile bildirebilirsiniz.

Diyelim ki `name` değerine sahip `Person` sınıfınız var:

```
Python hl_lines="1-3" {!../../../docs_src/python_types/tutorial010.py!}
```

Sonra bir değişkeni 'Person' tipinde tanımlayabilirsiniz:

```
Python hl_lines="6" {!../../../docs_src/python_types/tutorial010.py!}
```

Ve yine bütün editör desteğini alırsınız:

Pydantic modelleri

Pydantic veri doğrulaması yapmak için bir Python kütüphanesidir.

Verilerin "biçimini" niteliklere sahip sınıflar olarak düzenlersiniz.

Ve her niteliğin bir türü vardır.

Sınıfın bazı değerlerle bir örneğini oluşturursunuz ve değerleri doğrular, bunları uygun türe dönüştürür ve size tüm verileri içeren bir nesne verir.

Ve ortaya çıkan nesne üzerindeki bütün editör desteğini alırsınız.

Resmi Pydantic dokümanlarından alınmıştır:

```
{!../../../docs_src/python_types/tutorial011.py!}
```

!!! info Daha fazla şey öğrenmek için Pydantic'i takip edin.

FastAPI tamamen Pydantic'e dayanmaktadır.

Daha fazlasini görmek için Tutorial - User Guide.

FastAPI tip belirteçleri

FastAPI birkaç şey yapmak için bu tür tip belirteçlerinden faydalanır.

FastAPI ile parametre tiplerini bildirirsiniz ve şunları elde edersiniz:

- **Editor desteği**.
- **Tip kontrolü**.

... ve **FastAPI** aynı belirteçleri şunlar için de kullanıyor:

- **Gereksinimleri tanımlama**: request path parameters, query parameters, headers, bodies, dependencies, ve benzeri gereksinimlerden
- **Verileri çevirme**: Gönderilen veri tipinden istenilen veri tipine çevirme.
- **Verileri doğrulama**: Her gönderilen verinin:
  - doğrulanması ve geçersiz olduğunda **otomatik hata** oluşturma.
- OpenAPI kullanarak apinizi **Belgeleyin** :

- bu daha sonra otomatik etkileşimli dokümantasyon kullanıcı arayüzü tarafından kullanılır.

Bütün bunlar kulağa soyut gelebilir. Merak etme. Tüm bunları çalışırken göreceksiniz. Tutorial - User Guide.

Önemli olan, standart Python türlerini tek bir yerde kullanarak (daha fazla sınıf, dekoratör vb. eklemek yerine), **FastAPI**'nin bizim için işi yapmasını sağlamak.

!!! info Tüm öğreticiyi zaten okuduysanız ve türler hakkında daha fazla bilgi için geri döndüyseniz, iyi bir kaynak: the "cheat sheet" from `mypy`.