Uygulamalı Laboratuvar: Nose ile Test Çalıştırma

Gerekli tahmini süre: 30 dakika

Nose ile Test Çalıştırma laboratuvarına hoş geldiniz. Bu laboratuvar, Python'da birim testlerini çalıştırmak için temel araçları nasıl kullanacağınızı öğreneceksiniz.

Öğrenme Hedefleri

Bu laboratuvarı tamamladıktan sonra şunları yapabileceksiniz:

- Nose, Pinocchio ve Coverage'ı kurmak
- unittest ve Nose ile birim testlerini çalıştırmak
- Renk kodlu test çıktısı üretmek
- Test çıktınıza kapsam raporları eklemek

Theia Hakkında

Theia, masaüstünde veya bulutta çalıştırılabilen açık kaynaklı bir IDE (Entegre Geliştirme Ortamı)dir. Bu laboratuvarı yapmak için Theia IDE'sini kullanacaksınız. Theia ortamına giriş yaptığınızda, yalnızca sizin için ayrılmış bir 'bulut üzerindeki bilgisayar' ile karşılaşırsınız. Laboratuvarlarla çalıştığınız sürece bu size açıktır. Çıkış yaptığınızda, bu 'bulut üzerindeki bilgisayar' ve oluşturduğunuz dosyalar silinir. Bu nedenle, laboratuvarlarınızı tek bir oturumda tamamlamak iyi bir fikirdir. Laboratuvarın bir kısmını bitirir ve daha sonra Theia laboratuvarına dönerseniz, baştan başlamanız gerekebilir. Tüm Theia laboratuvarlarınızı tamamlamak için yeterli zamanınız olduğunda çalışmayı planlayın.

Laboratuvar Ortamını Kurma

Laboratuvara başlamadan önce biraz hazırlık yapmanız gerekiyor.

Editördeki menüyü kullanarak bir terminal penceresi açın: Terminal > Yeni Terminal.

Terminalde, eğer zaten `/home/projects` klasöründe değilseniz, şimdi proje klasörünüze geçin.

```
cd /home/project
```

Git Deposu Klonlama

Şimdi test etmeniz gereken kodu alalım. Bunu yapmak için, git deposunu klonlamak için `git clone` komutunu kullanacaksınız:

```
git clone https://github.com/ibm-developer-skills-network/duwjx-tdd_bdd_PracticeCode.git
```

Your output should look similar to the image below:

Laboratuvar Klasörüne Geçin

Depoyu klonladıktan sonra, `duwjx-tdd_bdd_PracticeCode` adlı dizine geçin.

```
cd duwjx-tdd_bdd_PracticeCode
```

İlk laboratuvar setine girmek için `labs/01_running_tests_with_nose/` dizinine geçin:

```
cd labs/01_running_tests_with_nose/
```

Bu dizinin içeriğini listeleyerek bu laboratuvar için oluşturulan eserleri görebilirsiniz.

```
ls -l
```

Klasör aşağıdaki listeye benzer görünmelidir:

Not: *İzleyeceğiniz adımlarda çalıştıracağınız birkaç alıştırma dosyanız olmalıdır.*

Ayrıca, dosyaları dosya gezgini içinde de görüntüleyebilirsiniz.

Adım 1: unittest ile Çalışma

İlk testlerinizi çalıştırmaya hazırsınız. `unittest` modülü Python 3'te yerleşiktir. Bunu Python yorumlayıcısı aracılığıyla, bir modülü çalıştırmak için `-m` argümanını ve ardından çalıştırmak istediğiniz modülün adını vermek için `unittest` kullanarak çağırabilirsiniz.

Terminalde, Python `unittest` paketini kullanarak testleri çalıştırmak için aşağıdaki komutu çalıştırın:

```
python3 -m unittest
```

Çıktıda, 11 nokta ile gösterildiği gibi tüm 11 testin geçtiğini göreceksiniz. Eğer herhangi bir hata olsaydı, noktaların yerinde `E` harfini görürdünüz; bu, o testte bir hata olduğunu temsil eder.

Daha ayrıntılı çıktı

Çıktıyı daha yararlı hale getirmek için `-v` bayrağını ekleyerek ayrıntılı modu açabilirsiniz. Daha yararlı bir çıktı için aşağıdaki komutu çalıştırın:

```
python3 -m unittest -v
```

Ayrıca, ayrıntılı modun test fonksiyon adı ve ardından gelen dokümantasyon dizesi gibi birçok tekrar eden bilgi verdiğini unutmayın. Sonraki bölümde `nose`'un bunu nasıl ele aldığını göreceksiniz.

Adım 2: Nose ile Çalışma

Daha iyi test çıktısı üretmek için kullanabileceğiniz bir test çalıştırıcısı olan **nose** vardır. Bu, Python paket yöneticisi `pip` aracıyla yükleyebileceğiniz bir Python paketidir.

Nose'u `pip` kullanarak yükleyin:

```
python3.8 -m pip install nose
```

Not: *Nose hakkında hafızanızı tazelemek için, Nose ile Testler bölümünü gözden geçirin.*

`nose`'dan ayrıntılı çıktı görmek için `nosetests -v` komutunu çalıştırın. Nose'dan alınan ayrıntılı çıktı, yalnızca docstring yorumlarını döndürdüğü için unittest'ten daha güzel bir çıktı verir:

```
nosetests -v
```

Adım 3: Pinocchio ile renk ekleme

Çıktınızı daha güzel hale getirmenin bir diğer yolu `pinocchio` adlı bir eklentiyi kullanmaktır. Bu eklenti ile çıktınızı **Rspec** benzeri bir spesifikasyon olarak alabilir ve ayrıca çıktınıza renk ekleyebilirsiniz. Renk, TDD'nin ünlü olduğu Kırmızı/Yeşil/Dönüştürme iş akışını gerçekten yansıtır.

`pip` kullanarak `pinocchio`'yu kurun.

```
python3.8 -m pip install pinocchio
```

Daha güzel bir biçimlendirme ve renkli bir çıktı almak için `nose`'u tekrar çalıştırın ve `--with-spec --spec-color` parametrelerini ekleyin:

```
nosetests --with-spec --spec-color
```

> Not: Çıktıda yeşil renk, tüm testlerin geçtiğini gösterir. Herhangi bir test başarısız olursa, o testin çıktısındaki rengi kırmızı olacaktır. Ayrıca, artık `-v`'ye ihtiyacınız yok çünkü `--with-spec` zaten ayrıntılı çıktı verir.

Adım 4: Test Kapsamı Ekleme

Yeterince testiniz olup olmadığını bilmek için, testlerinizin kaç satır kodu kapsadığını bilmeniz gerekir. `coverage` aracı, testleriniz sırasında yürütülen kod satırlarının sayısını toplam kod satırlarıyla karşılaştırarak, bunu bir kapsam yüzdesi olarak raporlayacaktır.

Test kapsamınızı kontrol edebilmek için `coverage` aracını kurun:

```
python3.8 -m pip install coverage
```

Sonra, `--with-coverage` parametresini ekleyerek `nose` üzerinden `coverage` çağırın.

```
nosetests --with-spec --spec-color --with-coverage
```

Adım 5: Eksik kapsama raporu oluştur

Kapsama aracının yararlı bir özelliği, hangi kod satırlarının eksik kapsama olduğunu raporlayabilmesidir. Bu bilgiyle, testlerinizi o eksik kod satırlarını çalıştıracak şekilde daha fazla test durumu eklemeniz gereken satırları bilirsiniz.

Eksik kapsama raporunu almak için terminalde aşağıdaki komutu çalıştırın:

```
coverage report -m
```

Bu test durumları %100 kapsama sağlar, ancak **Eksik** başlığıyla yeni bir sütunun eklendiğini fark edin. Bu, test kapsamı olmayan satırlar için satır numarlarının gösterileceği yerdir.

Adım 6: Bu parametreleri otomatikleştirme

Şimdiye kadar, testleri çalıştırırken birçok komut parametresi yazdınız. Alternatif olarak, tüm parametreleri bir yapılandırma dosyasında kaydedebilir ve her seferinde yazmak zorunda kalmazsınız.

`duwjx-tdd_bdd_PracticeCode/labs/01_running_tests_with_nose` dizininde `setup.cfg` adında yeni bir dosya oluşturun. İşte adımlar:

1. Sağdaki pencerede, Dosya menüsüne tıklayın ve aşağıdaki resimde gösterildiği gibi Yeni Dosya seçeneğini seçin:

2. Aşağıdaki resimde gösterildiği gibi Yeni Dosya başlıklı bir açılır pencere belirecektir. Dosya adı olarak `setup.cfg` girin ve ardından Tamam'a tıklayın.

3. Sizin için bir `setup.cfg` dosyası oluşturulacaktır.

Open **setup.cfg** in IDE

4. Aşağıdaki kodu `setup.cfg` dosyasına kopyalayıp yapıştırın:

```
[nosetests]
verbosity=2
with-spec=1
spec-color=1
with-coverage=1
```

```
cover-erase=1
cover-package=triangle
[coverage:report]
show_missing = True
```

setup.cfg dosyasını kaydetmeyi unutmayın.

Adım 7: Yapılandırma ile nosetests'i çalıştır

Artık `setup.cfg` dosyanızı oluşturduğunuza göre, terminalinize gidin ve `nosetests` komutunu herhangi bir parametre olmadan çalıştırın:

```
nosetests
```

Şimdi tüm parametrelerinizi ayarladınız ve sadece **nosetests** çalıştırarak renkli çıktı aldınız.

Sonuç

Testleri Nose ile Çalıştırma Laboratuvarını Tamamladığınız İçin Tebrikler

Artık Nose ile temel ve ayrıntılı birim testlerini nasıl çalıştıracağınızı biliyorsunuz. Ayrıca, Pinocchio ve Coverage eklentilerini kullanarak daha güzel test çıktıları üretmeyi ve Kırmızı/Yeşil/Refaktör iş akışını gerçekten hayata geçirmeyi de öğrendiniz. Bu komutlarla biraz daha deneme yapabilir veya bir sonraki derse geçebilirsiniz.

Artık bu araçları kendi projelerinizde uygulanabilir test durumu raporları üretmek için kullanmalısınız.

Author(s)

John J. Rofrano

Contributor(s)

Srishti Srivastava