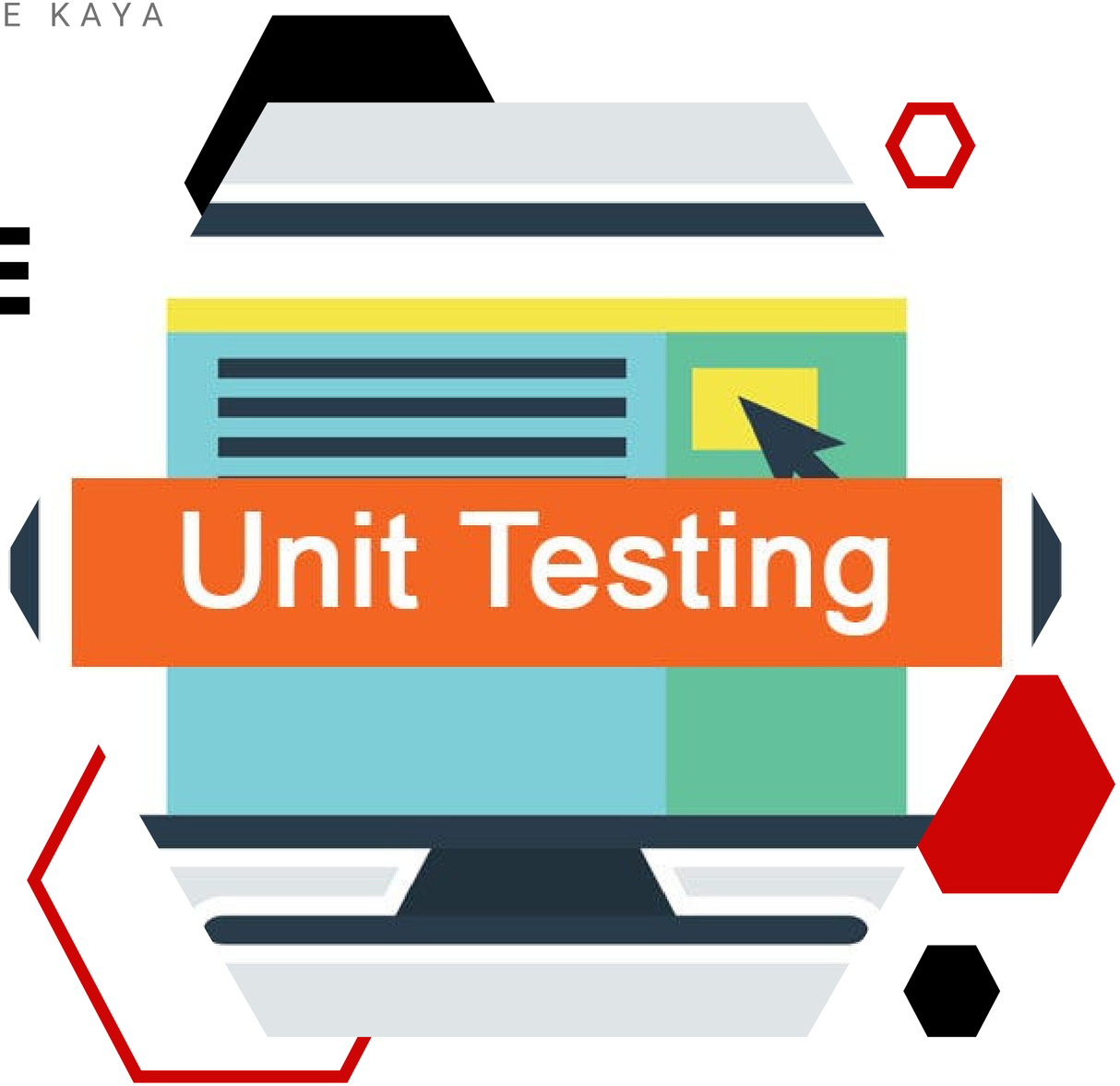
PROF. DR. BILAL ALATAŞ ARŞ. GÖR. ŞULE KAYA

YAZILIM GEÇERLEME VE SINAMA



- **2. HAFTA**
  - BİRİM TESTİ NEDİR?
  - BİRİM TESTİ NASIL YAPILIR?

Ders İçeriği

1. **Birim Testi (Unit Testing)**
2. Entegrasyon Testi (Integration Testing)
3. Sistem Testi (System Testing)
4. Kabul Testi (Acceptance Testing)
5. Regresyon Testi (Regression Testing)
6. Duman Testi (Smoke Testing)
7. Performans Testi
8. Kullanılabilirlik Testi (Usability Testing)
9. Güvenlik Testi (Security Testing)
10. Beta Testi

Ders İçeriği

Bu derste verilecek tüm örnekler **Python** dili ile gerçekleştirilecektir. Ek olarak diğer dillerde de kod örnekleri verilecektir. Ancak işleyiş Python üzerinden olacaktır.

Kodların paylaşımı ve çalıştırılması kolaylığından dolayı **Google Colab** isimli bulut tabanlı Jupyter Notebook ortamı kullanılacaktır. Öğrenciler istedikleri IDE ve geliştirme araçlarını kullanmakta özgürdürler.

Kısaca Hatırlayalım

Test Nedir?

Ana uygulamadan bağımsız, uygulamanın belirli kısımlarının farklı durumlarda hata üretip üretmediğini ya da alınan sonuçların normal veya doğru olup olmadığını kontrol etme pratiğine testing denilmektedir. Testler “manuel” ve “otomatik” olarak ikiye ayrılır. Bu derste, otomatik test ile ilgilenilecektir.

Testler ne tür hatalar yakalayabilir?

Fonksiyonel hatalar, veri hataları, arayüz ve erişilebilirlik hataları, Performans Hataları, Güvenlik Açıkları, Hesaplama ve Lojik Hataları, Yazılım Hataları, ve Entegrasyon Hataları olmak üzere bir çok hatayı yakalayabilir.

Kısaca Hatırlayalım

```python
def topla(x, y):
    return x - y  # Hata: Toplama işlemi yerine çıkarma işlemi yapılıyor.

# Test
assert topla(5, 3) == 8, f"Hata: Beklenen sonuç 8, alınan sonuç {topla(5, 3)}"
```

Bu örnekte, topla fonksiyonunun toplama yerine çıkarma işlemi yaptığı bir hata bulunmaktadır. Test, bu hatayı assert ifadesi ile tespit eder ve hata mesajını döndürür.

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-3-6944ad07001f> in <cell line: 5>()
      3
      4 # Test
----> 5 assert topla(5, 3) == 8, f"Hata: Beklenen sonuç 8, alınan sonuç {topla(5, 3)}"

AssertionError: Hata: Beklenen sonuç 8, alınan sonuç 2
```

Birim Testi (Unit Test) Nedir?

Birim testi (Unit Testing), yazılım geliştirme sürecinde kodun doğru ve beklenildiği gibi çalışıp çalışmadığını kontrol etmek için kullanılan bir test türüdür.

Birim testleri, genellikle bir fonksiyon veya metodun belirli bir girdi seti için beklenen çıktıyı verip vermediğini kontrol eder. Python'da birim testi yapmak için unittest adında bir kütüphane bulunur.

Birim Testi (Unit Test) Örnek

Python'da unittest kütüphanesi kullanılarak birim testi yapılabilmektedir. Aşağıda basit bir örnek verilmiştir:

```python
import unittest

# Test edilecek fonksiyon
def topla(x, y):
    return x + y

# Test sınıfı
class TestToplama(unittest.TestCase):

    def test_topla_pozitif(self):
        self.assertEqual(topla(5, 3), 8)  # Beklenen sonuç nedir?

    def test_topla_negatif(self):
        self.assertEqual(topla(-5, -3), -8)  # Beklenen sonuç nedir?

# Testlerin çalıştırılması
if __name__ == '__main__':
    unittest.TextTestRunner().run(unittest.TestLoader().loadTestsFromTestCase(TestToplama))
```

Birim Testi (Unit Test) Örnek

Bu kod, topla adlı bir fonksiyon tanımlar ve bu fonksiyonun doğruluğunu test etmek için TestToplama adında bir test sınıfı oluşturur. Test sınıfı içinde, pozitif ve negatif sayılarla toplama işlemleri test edilir ve beklenen sonuçlarla karşılaştırılır. Eğer script doğrudan çalıştırılırsa, unittest.TextTestRunner() kullanılarak bu testler çalıştırılır ve sonuçlar ekrana yazdırılır.

```
..
----------------------------
Ran 2 tests in 0.012s

OK
```

Birim Testi Nasıl Yapılır?

```
def validate_email(address):
    import re
    return bool(re.match(r'[^@]+@[^@]+\.[^@]+', address))

import unittest

class ValidatorsTestCase(unittest.TestCase):
    def test_validate_email(self):
        self.assertTrue(validate_email('birisi@gmail.com'))

if __name__ == '__main__':
    unittest.TextTestRunner().run(unittest.TestLoader().loadTestsFromTestCase(ValidatorsTestCase))
```

Birim Testi Nasıl Yapılır?

Birim testlerimi yapmak için python standart kütüphanesindeki unittest modülünü kullanılmaktadır. Burada dikkat edilmesi gereken; test sınıfının unittest.TestCase sınıfından miras alıyor olması ve test methodlarının isminin test ile başlıyor olmasıdır. Aksi halde bu fonksiyonlar test olarak görülmeyip, ilgili kısımlar teste tabi tutulmaz. Ancak hata da almayız.

Testi çalıştırdığınızda ”Ran 1 test in 0.000s” benzeri bir çıktı aldığınızda kaç tane testin çalıştırıldığını ve ne kadar zaman aldığını görebiliriz.

Miras aldığımız TestCase sınıfının assertTrue methodu, verilen parametrenin True olup olmadığını kontrol eder.

Birim Testi - Assert Metodları

Method	Checks that	New in
`assertEqual(a, b)`	`a == b`	
`assertNotEqual(a, b)`	`a != b`	
`assertTrue(x)`	`bool(x) is True`	
`assertFalse(x)`	`bool(x) is False`	
`assertIs(a, b)`	`a is b`	3.1
`assertIsNot(a, b)`	`a is not b`	3.1
`assertIsNone(x)`	`x is None`	3.1
`assertIsNotNone(x)`	`x is not None`	3.1
`assertIn(a, b)`	`a in b`	3.1
`assertNotIn(a, b)`	`a not in b`	3.1
`assertIsInstance(a, b)`	`isinstance(a, b)`	3.2
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`	3.2

https://docs.python.org/3/library/unittest.html#assert-methods adresinden, tüm assert metodlarının ne işe yaradığını öğrenebilirsiniz.

Birim Testi Örneği

İlk olarak, email-validator kütüphanesini yükleyelim:

```
!pip install email-validator
```

Birim Testi Örneği

```python
import unittest
from email_validator import validate_email, EmailNotValidError

def is_valid_email(address):
    try:
        # validate_email fonksiyonu bir hata fırlatır eğer e-posta geçerli değilse
        validate_email(address)
        return True
    except EmailNotValidError:
        return False

class ValidatorsTestCase(unittest.TestCase):
    def test_validate_email(self):
        # Bu format kesinlikle doğru.
        self.assertTrue(is_valid_email('birisi@gmail.com'))

        # Alt tire, tire ya da rakam içerebilir.
        self.assertTrue(is_valid_email('b---i_risi32@gmail.com'))

        # Soru işareti ya da ünlem içeremez.
        self.assertFalse(is_valid_email('birisi?!@gmail.com'))

        # Mutlaka @ işareti içermelidir.
        self.assertFalse(is_valid_email('birisigmail.com'))

        # @ işareti adresin başında olamaz.
        self.assertFalse(is_valid_email('@birisigmail.com'))

        # @ işaretinden bir adet nokta içermelidir.
        self.assertFalse(is_valid_email('birisii@gmailcom'))

if __name__ == '__main__':
    unittest.TextTestRunner().run(unittest.TestLoader().loadTestsFromTestCase(ValidatorsTestCase))
```

Bu kodun çıktısı nedir?

Birim Testi Örneği

Bu hatada, ***ValidatorsTestCase*** sınıfındaki *test_validate_email* metodunun bir bölümünde bir hata olduğunu görüyoruz. Spesifik olarak, 'birisi?!@gmail.com' e-posta adresini test ederken, *is_valid_email* fonksiyonu **True** döndürüyor, ancak test metodumuz ***self.assertFalse(is_valid_email('birisi?!@gmail.com'))*** satırı ile **False** döndürmesini bekliyor. Bu nedenle, bu beklenti karşılanmadığı için bir **AssertionError** hatası alıyoruz.

```
F
======================================================================
FAIL: test_validate_email (__main__.ValidatorsTestCase)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "<ipython-input-4-75e7a04b4260>", line 21, in test_validate_email
    self.assertFalse(is_valid_email('birisi?ulan!@gmail.com'))
AssertionError: True is not false

----------------------------------------------------------------------
Ran 1 test in 0.030s

FAILED (failures=1)
```

Birim Testi Örneği

Bu durum, email-validator kütüphanesinin bu e-posta adresini geçerli olarak değerlendirdiğini ve bu nedenle ***is_valid_email*** fonksiyonunun **True** döndürdüğünü gösterir. Eğer bu e-posta adresinin geçersiz olduğunu düşünüyorsanız ve bu durumu test etmek istiyorsanız, farklı bir e-posta doğrulama kütüphanesi kullanmayı düşünebilir veya kendi doğrulama mantığınızı yazabilirsiniz.

Birim Testi Örneği

Örneğin, e-posta adreslerini doğrulamak için daha basit bir regex kullanarak kendi ***is_valid_email*** fonksiyonunuzu yazabilirsiniz. Ancak, bu durumda, e-posta doğrulama kütüphanesinin sağladığı daha kapsamlı doğrulama işlevselliğinden vazgeçmiş olursunuz.

Bu hata, testlerin ne kadar önemli olduğunu ve potansiyel hataları veya yanlış anlamaları erken bir aşamada yakalamanıza nasıl yardımcı olabileceğini gösterir. Ayrıca, farklı kütüphanelerin ve araçların, belirli girdiler için farklı davranışlar sergileyebileceğini ve bu nedenle her zaman kendi testlerinizi yazmanızın önemli olduğunu hatırlatır.

1. Haftanın Özeti

1. **Birim Testi (Unit Testing):**
   - Birim testi, kodun belirli bir bölümünün veya fonksiyonunun beklenen davranışı gösterip göstermediğini kontrol etmek için kullanılır.
   - Python'da **unittest** modülü, birim testlerini yazmak ve çalıştırmak için kullanılır.
2. **Test Sınıfı ve Miras:**
   - Test sınıfları, **unittest.TestCase** sınıfından türetilir.
   - Test metodları, test ile başlayan isimlere sahip olmalıdır.
3. **Test Metodları ve Doğrulamalar:**
   - *assertTrue* ve *assertFalse* gibi metodlar, belirli koşulları doğrulamak için kullanılır.
4. **Test Çalıştırma:**
   - *unittest.main()* veya *unittest.TextTestRunner()* kullanılarak testler çalıştırılır.
   - Google Colab'da *unittest.TextTestRunner()* kullanımı tercih edilir.

1. Haftanın Özeti

5. **E-posta Doğrulama:**
    - E-posta doğrulama için **email-validator** kütüphanesi kullanıldı.
    - **validate_email** fonksiyonu, e-posta adreslerinin doğru formatta olup olmadığını kontrol eder.
6. **Hata Yakalama:**
    - Testler, kodunuzda beklenmeyen davranışları veya hataları yakalamanıza yardımcı olur.
    - **AssertionError**, beklenen ve gerçek sonuçlar arasında bir uyumsuzluk olduğunda fırlatılır.
7. **Google Colab:**
    - Google Colab, interaktif bir Python programlama ortamıdır.
    - **unittest.main()** metodunun Google Colab'da çalışmayacağı ve bu nedenle **unittest.TextTestRunner()** kullanılması gerektiği belirtildi.

1. Haftanın Özeti

8. **Kütüphane İthalat Hataları:**
    - ImportError, belirli bir kütüphanenin veya modülün bulunamadığını belirtir.
    - Kütüphane ithalat hataları, genellikle kütüphanenin yüklenmemiş olmasından veya yanlış bir isimle ithal edilmesinden kaynaklanır.

1. Hafta Ödevi

Hata Yakalama Testleri:

- Hatalı girdilerle hata durumlarını test eden birim testleri yazın.
- Fonksiyonunuzun beklenen hataları doğru bir şekilde fırlattığından emin olun.

NOT: Verilen ödevi 3. haftaya kadar sisteme yükleyiniz.