Nesneye Yönelimli Programlama – BLM 205

Hafta 13: Yazılım Testi

Fenerbahçe Üniversitesi

Öğretim Elemanları

Öğretim Üyesi: Dr. Vecdi Emre Levent

Ofis: 311

Email: emre.levent@fbu.edu.tr

Asistan: Arş. Gör. Uğur Özbalkan

Ofis: 307

Email: ugur.ozbalkan@fbu.edu.tr

Asistan: Arş. Gör. Ecenur Alioğulları

Ofis: 307

Email: ecenur.aliogullari@fbu.edu.tr

Ders Planı

- Yazılım Testi
  - Test yaklaşımları
  - Birim testi (Unit test)

Yazılım Testi

Hem hod geliştirme süreci devam ederken ve hemde tamamlandıktan sonra test edilmesi çok önemlidir.

İyi test edilmemiş yazılımlarda öngörülemeyen bir çok sorun ortaya çıkabilir.

Yazılım Testi

Bir çok test yapma yaklaşımı bulunabilir.

En ilkel yaklaşım, uygulamayı çalıştırıp çıktılarına bakmaktadır.

Debugger araçları ile kod yürütülürken daha detaylı analizler yapılabilir.

Takımdan bir başka kişiye, kod teslim edilerek incelenmesi istenebilir.

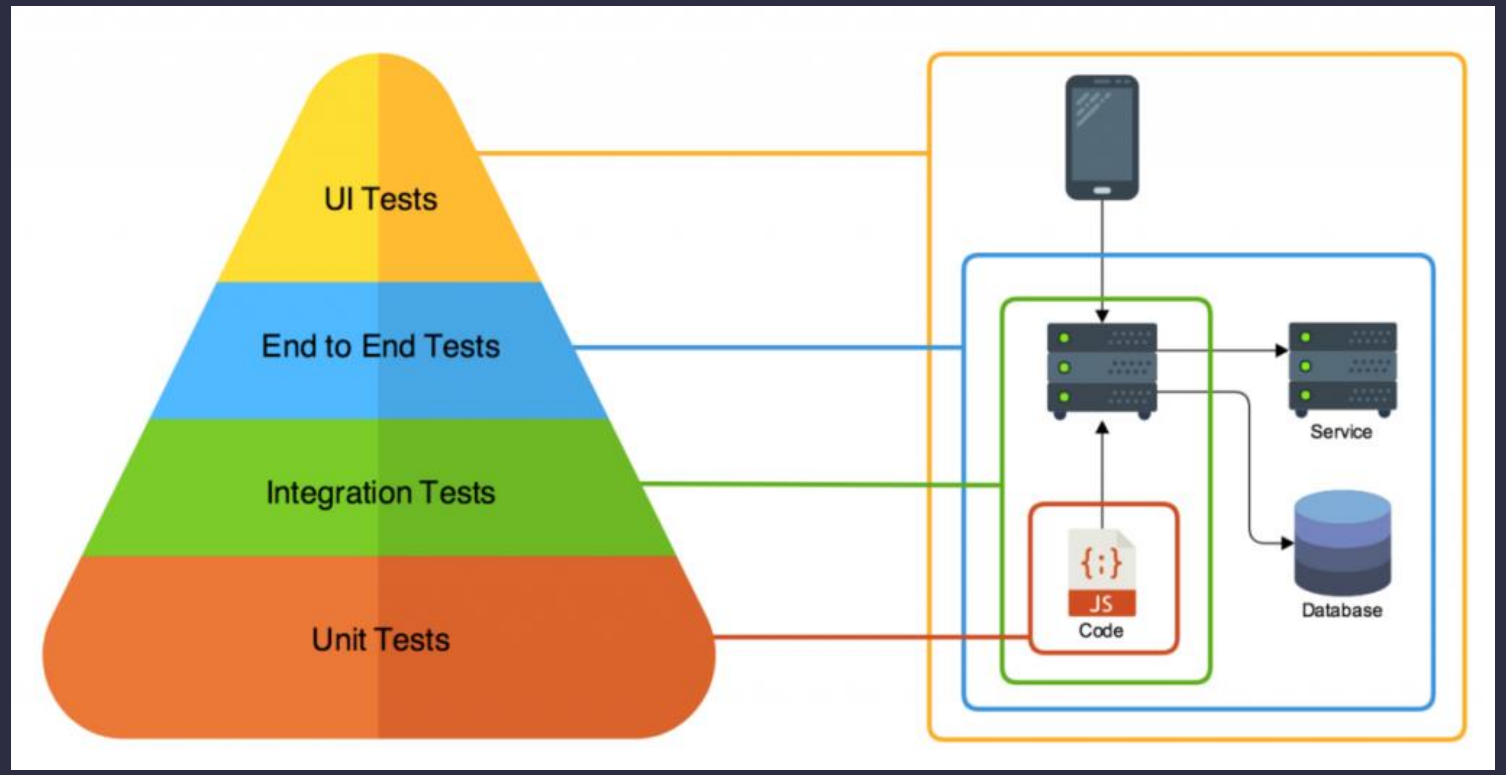
Yazılım Testi

Bu yaklaşımlar manual olarak adlandırılır.

Otomatize bir test yaklaşımı değildir.

Yazılım Testi

Otomatize test yaklaşımı için Unit test denilen teknik sıklıkla tercih edilmektedir.



Yazılım Testi

Örnek 1

isimFonksiyonu.py

```
def tamIsim(ad, soyad):
    tamIsimStr = ad + ' ' + soyad
    return tamIsimStr
```

main.py

```
from isimFonksiyonu import tamIsim

ad = input("Isim girin: ")
soyad = input("Soyisim girin: ")

sonuc = tamIsim(ad, soyad)
print("Tam isim: " + sonuc)
```

Manual olarak çıktıya bakıp test yapıldı

Çıktı

```
Isim girin: emre
Soyisim girin: levent
Tam isim: emre levent
```

Yazılım Testi

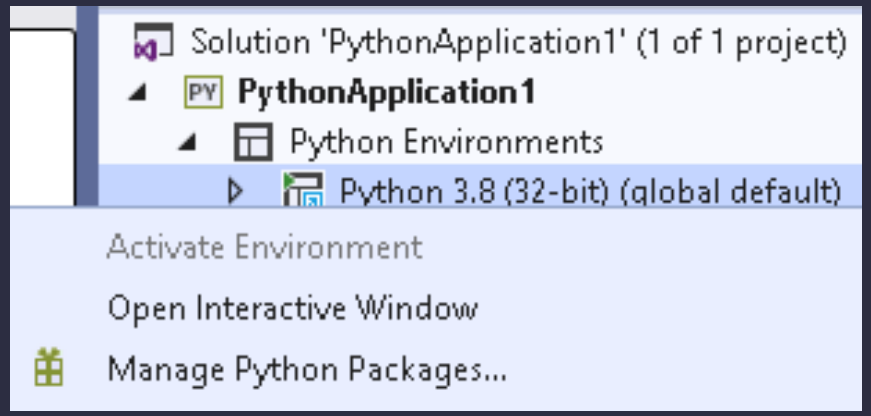
Otomatize test yapmak için;

- Bir test dosyası oluşturulmalıdır.
- Python'un bir modülü olan unittest modülü eklenmelidir.
- Test senaryoları kodlanmalıdır.

Yazılım Testi

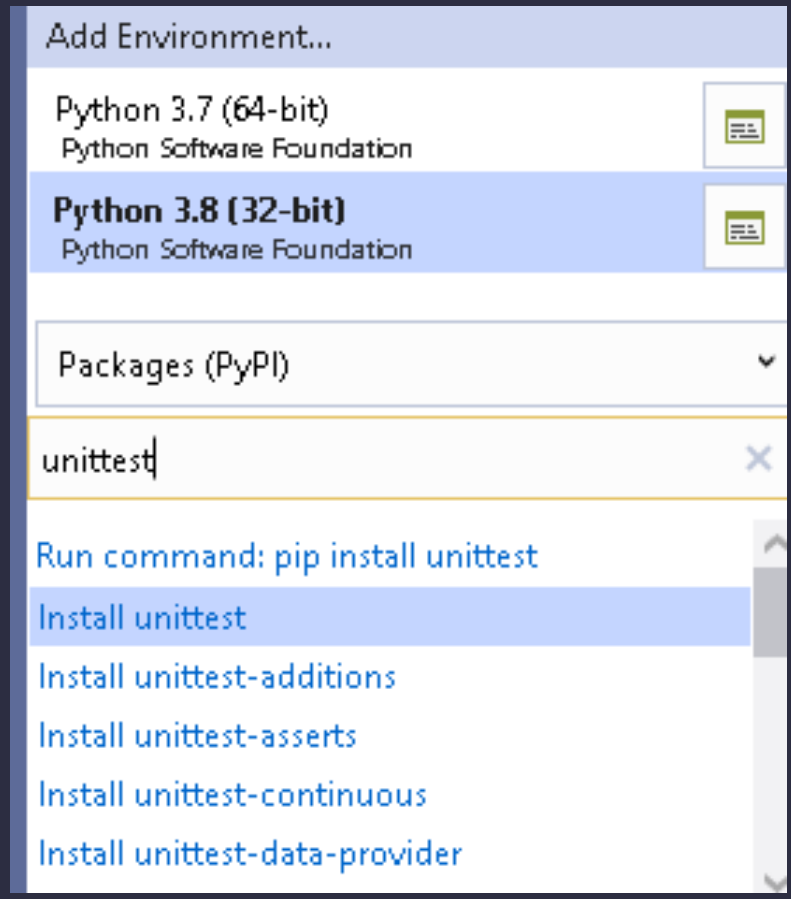
Visual Studio'da Python'a paket eklemek için, Python environments genişletilip, kullanılan python versiyonu seçilerek sağ tıklanır.

Açılan menüden manage Python Packages…'a tıklanarak, paket yükleme penceresi açılır.



Yazılım Testi

Açılan ekrandan istenen paket aranarak, bulunursa install "paket adı" ‘na tıklanarak yüklenebilir.



Yazılım Testi

Örnek 2

isimFonksiyonu.py

```
def tamIsim(ad, soyad):
    tamIsimStr = ad + ' ' + soyad
    return tamIsimStr
```

test.py

```
import unittest
from isimFonksiyonu import tamIsim

class testSinifi(unittest.TestCase):

    def tamIsimTestFonksiyonu(self):
        sonuc = tamIsim("emre", "levent")
        self.assertEqual(sonuc, "emre levent")

test1 = testSinifi()
test1.tamIsimTestFonksiyonu()
```

assertEqual fonksiyonu test amaçlı kullanılmaktadır.

Uygulamada hata olmadığı için, test.py dosyası yürütüldüğünde çıktı vermeyecektir.

Yazılım Testi

Örnek 3

isimFonksiyonu.py

```
def tamIsim(ad, soyad):
    tamIsimStr = ad + ' ' + soyad
    return tamIsimStr
```

test.py

```
import unittest
from isimFonksiyonu import tamIsim

class testSinifi(unittest.TestCase):

    def tamIsimTestFonksiyonu(self):
        sonuc = tamIsim("emre", "levent")
        self.assertEqual(sonuc, "emre X levent")

test1 = testSinifi()
test1.tamIsimTestFonksiyonu()
```

Exception Unhandled

```
'emre levent' != 'emre lX event'
- emre levent
+ emre lX event
?     ++
```

Uygulamada hata oluşacaktır.

Yazılım Testi

Yeni test durumları ekleme

isimFonksiyonu.py

```
def tamIsim(ad, soyad):
    tamIsimStr = ad + ' ' + soyad
    return tamIsimStr

def tamIsim2(ad, soyad):
    tamIsimStr = ad + '*' + soyad
    return tamIsimStr

def tamIsim3(ad, soyad):
    tamIsimStr = ad + '#' + soyad
    return tamIsimStr
```

test.py

```
import unittest
import isimFonksiyonu

class testSinifi(unittest.TestCase):

    def tamIsimTestFonksiyonu(self):
        sonuc = isimFonksiyonu.tamIsim("emre", "levent")
        self.assertEqual(sonuc, "emre levent")

    def tamIsimTestFonksiyonu2(self):
        sonuc = isimFonksiyonu.tamIsim2("emre", "levent")
        self.assertEqual(sonuc, "emre*levent")

    def tamIsimTestFonksiyonu3(self):
        sonuc = isimFonksiyonu.tamIsim3("emre", "levent")
        self.assertEqual(sonuc, "emre#levent")

test1 = testSinifi()
test1.tamIsimTestFonksiyonu()
test1.tamIsimTestFonksiyonu2()
test1.tamIsimTestFonksiyonu3()
```

Uygulamada hata olmadığı için, test.py dosyası yürütüldüğünde çıktı vermeyecektir.

Yazılım Testi

Yeni test durumları ekleme

test.py

```
import unittest

assert sum([1, 2, 3]) == 4, "Sonuc 6 olmali!"
```

İfade doğru olmadığı zaman, hata verdirilmesini sağlamaktadır.

Yazılım Testi

Örnek 4

toplama.py

```
def topla(arg):
    toplam = 0
    for deger in arg:
        toplam += deger
    return toplam
```

Uygulamada hata olmadığı için, test.py dosyası yürütüldüğünde çıktı vermeyecektir.

test.py

```
import unittest

from toplamaDosyasi import toplama

class toplamaTestiSinifi(unittest.TestCase):
    def toplamaTesti(self):

        veri = [1, 2, 3]
        sonuc = toplama(veri)
        self.assertEqual(sonuc, 6)

test1 = toplamaTestiSinifi()
test1.toplamaTesti()
```

Yazılım Testi

Assertions (İddalar)

Metot	Karşılık gelen ifade
.assertEqual(a, b)	a == b
.assertTrue(x)	bool(x) doğru ise
.assertFalse(x)	bool(x) yanlış ise
.assertIs(a, b)	a ve b aynı ise
.assertIsNone(x)	x none ise
.assertIn(a, b)	a, b'de var ise

test.py

```
import unittest

class testSinifi(unittest.TestCase):
    def testFnc(self):
        self.assertEqual(3,3.0)

test1 = testSinifi()
test1.testFnc()
```

→ Hata vermez

test.py

```
import unittest

class testSinifi(unittest.TestCase):
    def testFnc(self):
        self.assertIs(3,3.0)

test1 = testSinifi()
test1.testFnc()
```

→ Hata verir

Yazılım Testi

Örnek 5

toplama.py

```
def topla(arg):
    toplam = 0
    for deger in arg:
        toplam += deger
    return toplam
```

test.py

```
import unittest
from fractions import Fraction
from toplamaDosyasi import toplama

class toplamaTesti(unittest.TestCase):
    def toplamaTesti1(self):
        veri = [1, 2, 3]
        sonuc = toplama(veri)
        self.assertEqual(sonuc, 6)

    def toplamaTesti2(self):
        veri = [Fraction(1, 4), Fraction(1, 4),
Fraction(2, 5)]
        sonuc = toplama(veri)
        self.assertEqual(sonuc, 1)

test1 = toplamaTesti()
test1.toplamaTesti1()
test1.toplamaTesti2()
```

Kesirli sayılar,
Örn 1/4

9/10 olarak
hata verecektir

Yazılım Testi

Örnek 6

dikdortgen.py

```
def alan(en, boy):
    return en * boy

def cevre(en, boy):
    return en * boy
```

Uygulama hata vermeyecektir, ancak dikkat edildiğinde, çevre fonksiyonu hatalıdır.

Hata'nın tespit edilememesinin nedeni, yeterince farklı girişler denenmemesidir.

test.py

```
import unittest
from dikdortgen import alan
from dikdortgen import cevre

class dikdortgenTesti(unittest.TestCase):
    def alanTesti(self):
        sonuc = alan(2, 2)
        self.assertEqual(sonuc, 4)

    def cevreTesti(self):
        sonuc = cevre(2, 2)
        self.assertEqual(sonuc, 4)

test1 = dikdortgenTesti()
test1.alanTesti()
test1.cevreTesti()
```

Yazılım Testi

Örnek 6

dikdortgen.py

```
def alan(en, boy):
    return en * boy

def cevre(en, boy):
    return en * boy
```

Yeni eklenen test case'i ile hata verecektir.

test.py

```
import unittest
from dikdortgen import alan
from dikdortgen import cevre

class dikdortgenTesti(unittest.TestCase):
    def alanTesti(self):
        sonuc = alan(2, 2)
        self.assertEqual(sonuc, 4)
        sonuc = alan(2, 3)
        self.assertEqual(sonuc, 6)

    def cevreTesti(self):
        sonuc = cevre(2, 2)
        self.assertEqual(sonuc, 4)
        sonuc = cevre(2, 3)
        self.assertEqual(sonuc, 5)

test1 = dikdortgenTesti()
test1.alanTesti()
test1.cevreTesti()
```