



**UNIVERZITET U SARAJEVU**

**FAKULTET ZA SAOBRAĆAJ I KOMUNIKACIJE**



**GRUPNI PROJEKAT IZ PREDMETA:**

**TESTIRANJE SOFTVERA I UPRAVLJANJE KVALITETOM**

**ZADATAK -**  
**GP 3:**

**Unit testiranje korištenjem Assert funkcija**

<b>Predmetni nastavnik:</b>	
<b>Asistent (saradnik):</b>	/

<b>Studenti:</b>	Nejla Belagoši, Ajla Guhdija, Nejra Pezo
<b>Brojevi indeksa:</b>	1103/II, 1091/II ,1075/II
<b>Usmjerenja:</b>	KIT
<b>Godina studija:</b>	II (druga) godina
<b>Rezultat rada:</b>	

Sarajevo, Datum: 11.01.2025

## SADRŽAJ

1.   assertIsInstance(a,b).....	1
1.1.   Test za metodu broj 1:.....	1
2.   assertIn(a,b).....	5
2.1.   Test za funkciju br. 2.....	5
3.   assertRegex(s,r).....	8
3.1.   Test za funkciju br. 3.....	8
4.   assertSetEqual(a,b).....	12
4.1.   Test za funkciju br. 4.....	12
POPIS PRILOGA .....	15

#	Assert metoda
1	assertIsInstance(a,b)
2	assertIn(a,b)
3	assertRegex(s,r)
4	assertSetEqual(a,b)

#	Naziv
Jezik	Python (v.3.13.1)
Framework	Unittest
IDE	PyCharm

*\*LOC baziran na fizičkom brojanju.*

## 1. assertIsInstance(a,b)

OPIS:

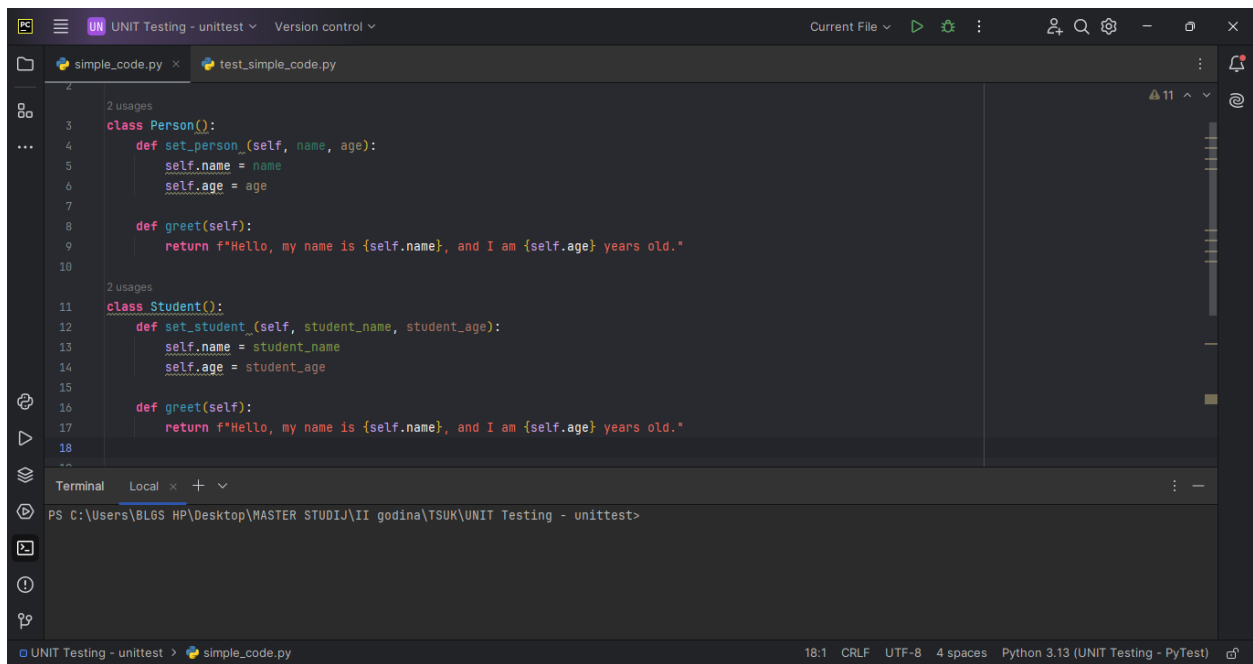
Metoda koja se koristi za provjeru je li objekt instanca date klase ili ne. Ova funkcija može uzeti u obzir 3 parametra kao ulaz prima: objekat koji se testira, klasa ili tuple klasa za provjeru, opcionalna poruka o grešci i ako je objekt instanca klase, test prolazi bez greške, ako nije, generiše se AssertionError.

### 1.1. Test za metodu broj 1:

Dvije klase: Person i Student

Metode: set\_person(), set\_student() i greet()

*Opis metoda:* set\_person() i set\_student() postavljaju ime i godine za osobu i studenta, greet() vraća tekstualnu poruku sa imenom i godinama osobe..



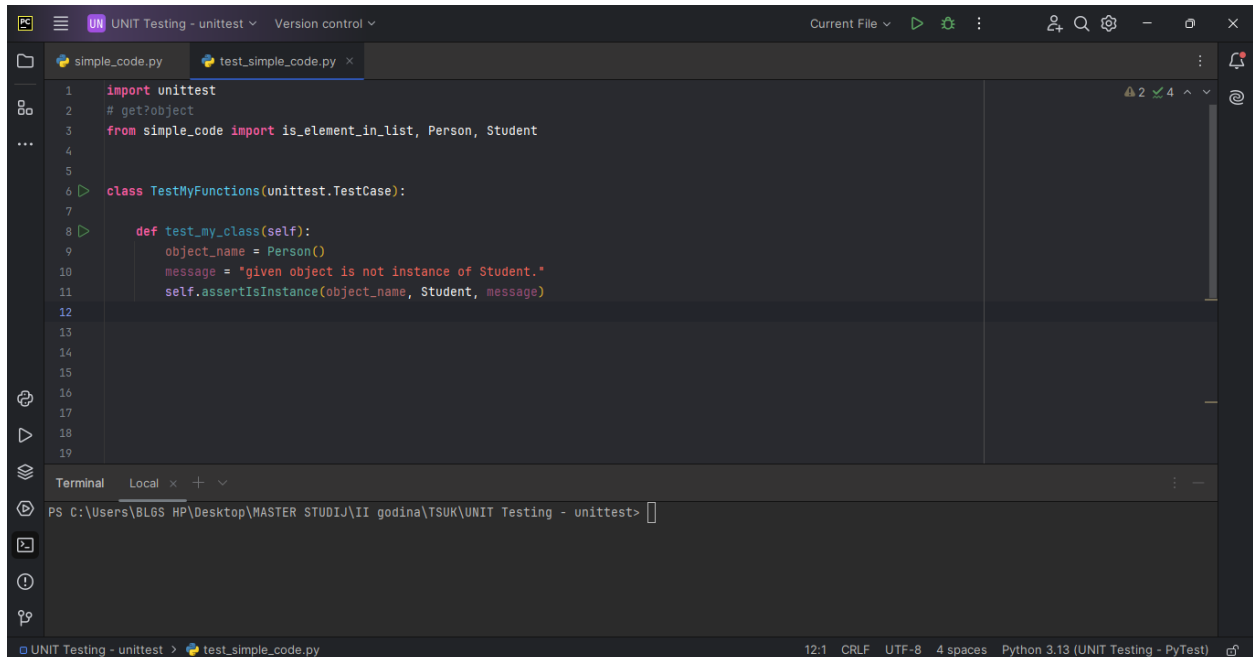
```
2 usages
3 class Person():
4     def set_person(self, name, age):
5         self.name = name
6         self.age = age
7
8     def greet(self):
9         return f"Hello, my name is {self.name}, and I am {self.age} years old."
10
11 2 usages
12 class Student():
13     def set_student(self, student_name, student_age):
14         self.name = student_name
15         self.age = student_age
16
17     def greet(self):
18         return f"Hello, my name is {self.name}, and I am {self.age} years old."
19
20
Terminal Local x + v
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJI\II godina\TSUK\UNIT Testing - unittest>
```

Slika 1 Primjer funkcije br. 1

### Primjer neuspješnog testa:

OPIS:

U ovoj metodi se provjerava da li je `object_name` instanca klase `Student`. Pošto je `object_name` instanca klase `Person`, `assertIsInstance()` će generisati `AssertionError` što će rezultovati neuspjeh testa.



The screenshot shows a code editor with a file named `test_simple_code.py`. The code defines a test class `TestMyFunctions` with a method `test_my_class`. Inside `test_my_class`, a `Person` object is created and passed to `assertIsInstance` along with the `Student` class and a message. The terminal at the bottom shows the command `python -m unittest test_simple_code.TestMyFunctions.test_my_class` being executed, but no output is visible yet.

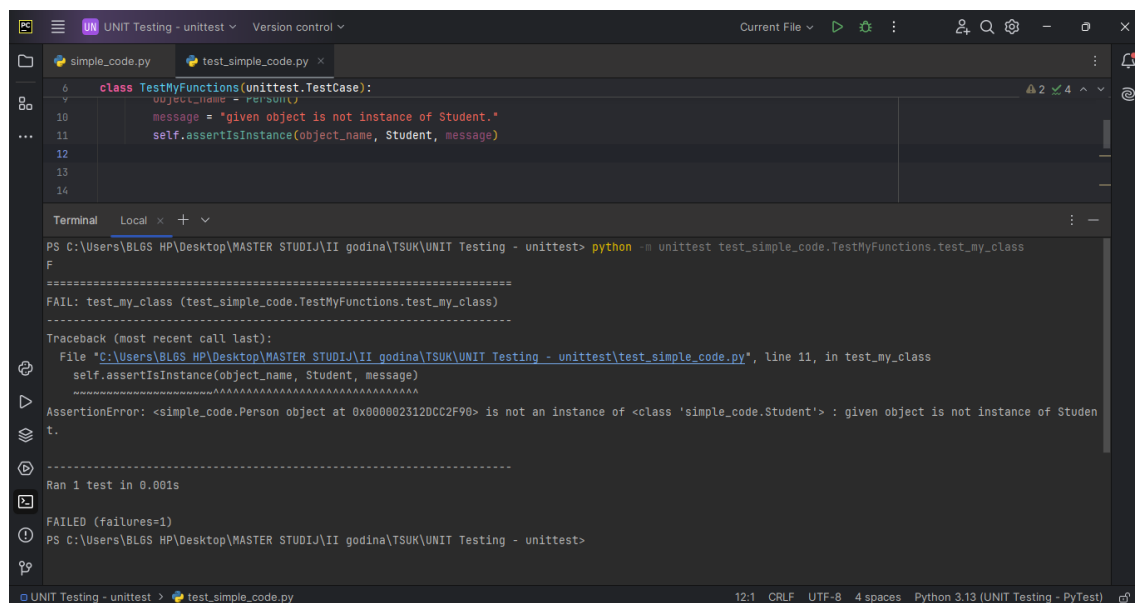
```
1 import unittest
2 # getObject
3 from simple_code import is_element_in_list, Person, Student
4
5
6 class TestMyFunctions(unittest.TestCase):
7
8     def test_my_class(self):
9         object_name = Person()
10        message = "given object is not instance of Student."
11        self.assertIsInstance(object_name, Student, message)
12
13
14
15
16
17
18
19
```

Terminal Local x + v

PS C:\Users\BLAGS HP\Desktop\MASTER STUDIJ\II godina\TSUK\UNIT Testing - unittest>

Slika 2 Primjer metode za neuspješni test funkcije br. 1

## Rezultat testa:



The screenshot shows the same code editor as before, but now the terminal displays the output of the test execution. The test fails with an `AssertionError`. The message indicates that the `Person` object is not an instance of the `Student` class.

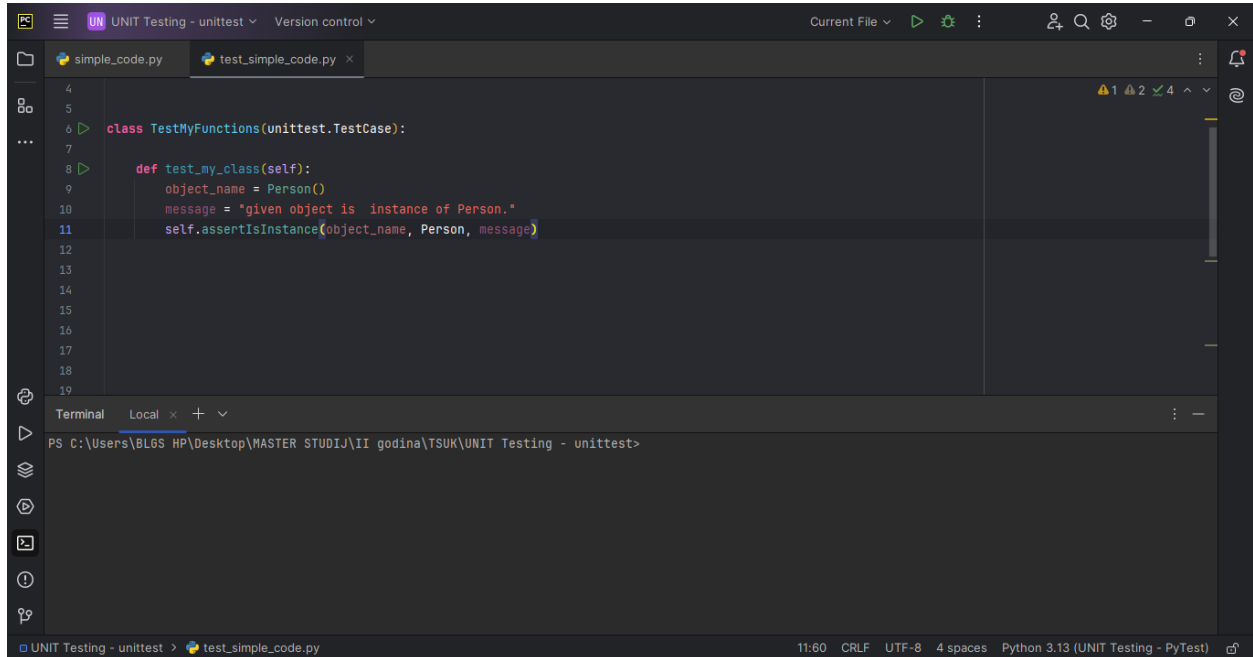
```
PS C:\Users\BLAGS HP\Desktop\MASTER STUDIJ\II godina\TSUK\UNIT Testing - unittest> python -m unittest test_simple_code.TestMyFunctions.test_my_class
F
=====
FAIL: test_my_class (test_simple_code.TestMyFunctions.test_my_class)
-----
Traceback (most recent call last):
  File "C:\Users\BLAGS HP\Desktop\MASTER STUDIJ\II godina\TSUK\UNIT Testing - unittest\test_simple_code.py", line 11, in test_my_class
    self.assertIsInstance(object_name, Student, message)
    ~~~~~^~~~~~
AssertionError: <simple_code.Person object at 0x0000023120CC2F90> is not an instance of <class 'simple_code.Student'> : given object is not instance of Student.
-----
Ran 1 test in 0.001s
FAILED (failures=1)
PS C:\Users\BLAGS HP\Desktop\MASTER STUDIJ\II godina\TSUK\UNIT Testing - unittest>
```

Slika 3 Rezultat neuspješnog testa za funkciju br. 1

## Primjer uspješnog testa:

### OPIS:

U ovoj metodi provjeravamo da li je `object_name` instanca klase `Person`, pošto `object_name` je kreiran kao instanca klase `Person`, test prolazi bez greške i metoda `assertIsInstance` će potvrditi instancu.



The screenshot shows a code editor with two tabs: `simple_code.py` and `test_simple_code.py`. The `test_simple_code.py` tab is active, displaying the following Python code:

```
4
5
6 class TestMyFunctions(unittest.TestCase):
7
8     def test_my_class(self):
9         object_name = Person()
10        message = "given object is instance of Person."
11        self.assertIsInstance(object_name, Person, message)
```

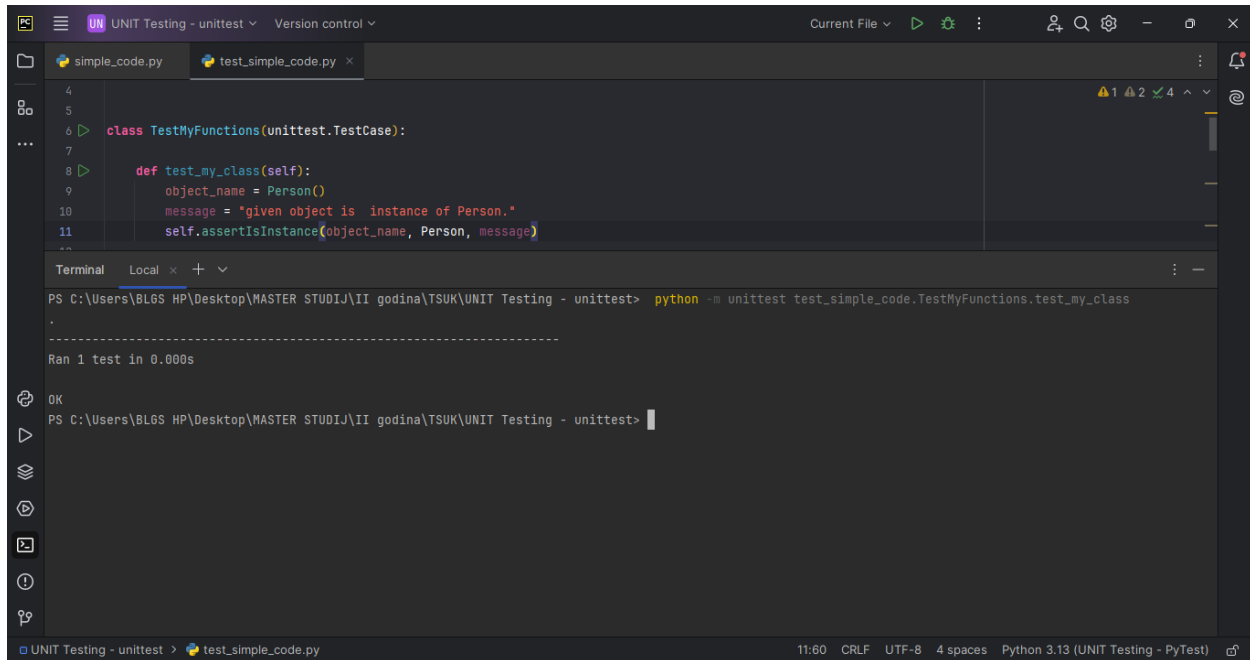
Below the code editor is a terminal window with the following command prompt:

```
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJA\II godina\TSUK\UNIT Testing - unittest>
```

The status bar at the bottom indicates the file is `test_simple_code.py`, the encoding is `UTF-8`, and the Python version is `Python 3.13 (UNIT Testing - PyTest)`.

Slika 4 Primjer metode za uspješni test za funkciju br. 1

## Rezultat testa:



The screenshot shows a code editor with two tabs: `simple_code.py` and `test_simple_code.py`. The `test_simple_code.py` tab is active, displaying the following Python code:

```
4  
5  
6 class TestMyFunctions(unittest.TestCase):  
7  
8     def test_my_class(self):  
9         object_name = Person()  
10        message = "given object is instance of Person."  
11        self.assertIsInstance(object_name, Person, message)  
..
```

Below the code editor is a terminal window. The command executed is:

```
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJI\II godina\TSUK\UNIT Testing - unittest> python -m unittest test_simple_code.TestMyFunctions.test_my_class
```

The output of the command is:

```
.....  
Ran 1 test in 0.000s  
  
OK  
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJI\II godina\TSUK\UNIT Testing - unittest>
```

The status bar at the bottom indicates the file is `test_simple_code.py`, the encoding is `UTF-8`, and the Python version is `Python 3.13 (UNIT Testing - PyTest)`.

Slika 5 Rezultat uspješnog testa metode

## 2. assertIn(a,b)

### OPIS:

Metoda koja se koristi za testiranje da li se određeni element (a) nalazi unutar sekvence ili kolekcije (b). Ako je element prisutan, test prolazi; u suprotnom, test ne uspijeva i generiše grešku.

### 2.1. Test za funkciju br. 2

#### Funkcija br. 2:

Naziv funkcije: *is\_element\_in\_list*

Opis funkcije: Funkcija *is\_element\_in\_list* kreira listu koja sadrži elemente u rasponu od n1 do n2 (uključujući oba kraja).

```
2 usages
def is_element_in_list(n1, n2):
    list = []
    for i in range(n1, n2+1):
        list.append(i)
    print(list)
    return list
```

Slika 6 Primjer funkcije br. 2

#### Primjer neuspješnog testa:

### OPIS:

U ovoj metodi se generiše lista elemenata u rasponu od 0 do 10, i zatim se provjerava da li se broj 34 nalazi u toj listi. Pošto u rasponu od 0 do 10, broj 34 se ne nalazi u listi, metoda *assertIn* izaziva neuspjeh testa.

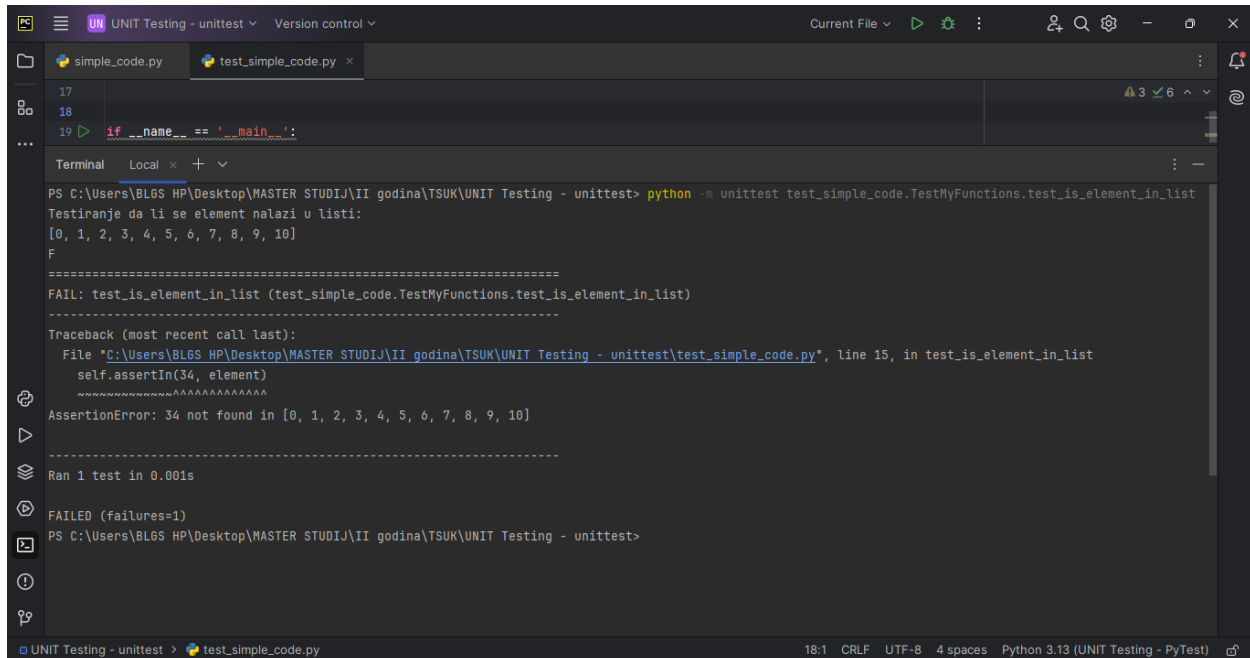
```
def test_is_element_in_list(self):
    print("Testiranje da li se element nalazi u listi:")
    element = is_element_in_list(n1: 0, n2: 10)
    self.assertIn(member: 34, element)

if __name__ == '__main__':
    unittest.main()
```

Slika 7 Primjer metode za neuspješni test za funkciju br. 2



## Rezultat testa:



```
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJA\II godina\TSUK\UNIT Testing - unittest> python -m unittest test_simple_code.TestMyFunctions.test_is_element_in_list
Testiranje da li se element nalazi u listi:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
F
=====
FAIL: test_is_element_in_list (test_simple_code.TestMyFunctions.test_is_element_in_list)
=====
Traceback (most recent call last):
  File "C:\Users\BLGS HP\Desktop\MASTER STUDIJA\II godina\TSUK\UNIT Testing - unittest\test_simple_code.py", line 15, in test_is_element_in_list
    self.assertIn(34, element)
AssertionError: 34 not found in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
-----
Ran 1 test in 0.001s

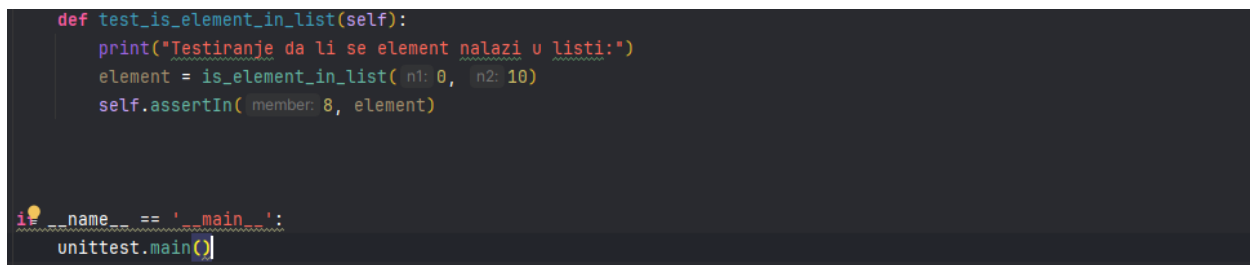
FAILED (failures=1)
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJA\II godina\TSUK\UNIT Testing - unittest>
```

Slika 8 Rezultat neuspješnog testa za funkciju br. 2

## Primjer uspješnog testa:

### OPIS:

U ovoj metodi se generiše lista brojeva u rasponu od 0 do 10, a zatim se provjerava da li se broj 8 nalazi u toj listi. Pošto broj 8 pripada ovom rasponu, metoda `assertIn` prolazi bez greške i test je uspješan.

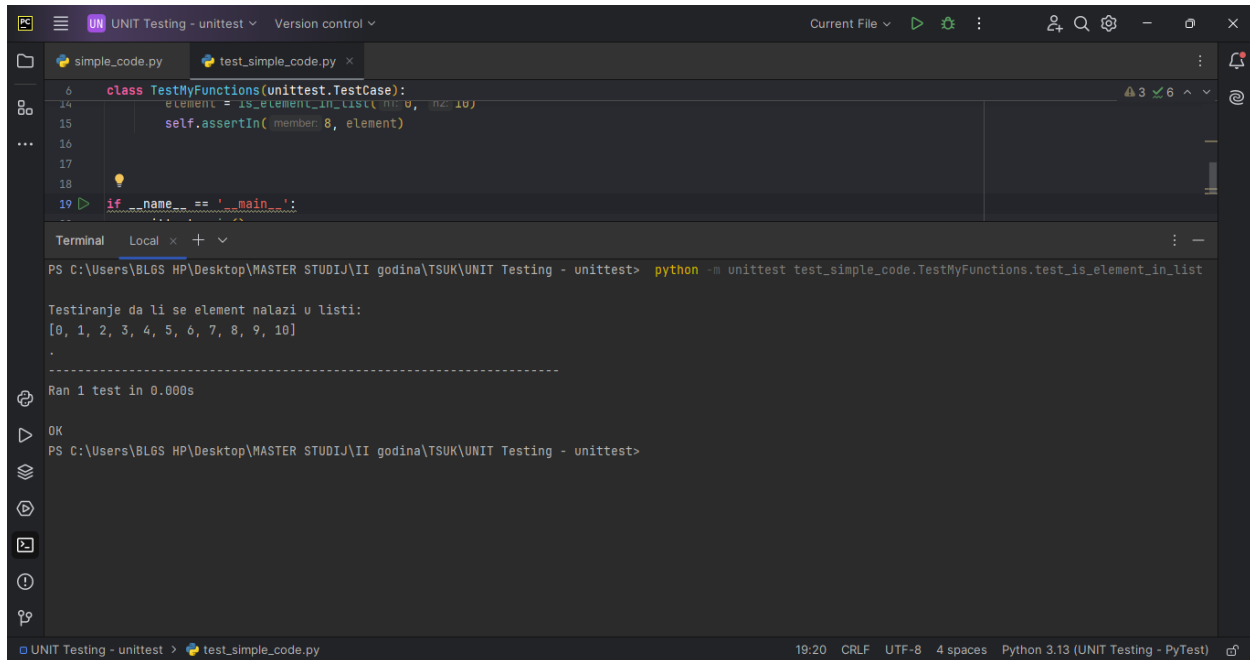


```
def test_is_element_in_list(self):
    print("Testiranje da li se element nalazi u listi:")
    element = is_element_in_list( n1: 0, n2: 10)
    self.assertIn( member: 8, element)

if __name__ == '__main__':
    unittest.main()
```

Slika 9 Primjer metode za uspješni test za funkciju br. 2

## Rezultat testa:



The screenshot shows a code editor with two tabs: 'simple\_code.py' and 'test\_simple\_code.py'. The 'test\_simple\_code.py' tab is active, displaying a Python class `TestMyFunctions` that inherits from `unittest.TestCase`. It contains a test method `test_is_element_in_list` that calls `is_element_in_list` with `member=8` and `element=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`. The test is marked as successful with a green checkmark. Below the code editor is a terminal window showing the command `python -m unittest test_simple_code.TestMyFunctions.test_is_element_in_list` and its output: 'Testiranje da li se element nalazi u listi: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]' followed by 'Ran 1 test in 0.000s' and 'OK'.

```
6 class TestMyFunctions(unittest.TestCase):
14     element = is_element_in_list([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 8)
15     self.assertIn(member=8, element)
16
17
18
19 if __name__ == '__main__':
    ...

Terminal Local x + v
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJ\II godina\TSUK\UNIT Testing - unittest> python -m unittest test_simple_code.TestMyFunctions.test_is_element_in_list

Testiranje da li se element nalazi u listi:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
.
-----
Ran 1 test in 0.000s

OK
PS C:\Users\BLGS HP\Desktop\MASTER STUDIJ\II godina\TSUK\UNIT Testing - unittest>
```

Slika 10 Rezultat uspješnog testa za funkciju br. 2

### 3. assertRegex(s,r)

#### OPIS:

Testiraju da li pretraga regularnim izrazom odgovara tekstu, odnosno da li string s odgovara regularnom izrazu r.

assertRegex(text, regex, msg=None)

- text -> tekstualni string koji se provjerava
- regex -> regularni string koji se koristi za provjeru,
- msg (opcionally) -> prilagođena poruka ako test ne uspije.

Ako ne postoji podudaranje, test pada i ne prikazuje grešku. Suprotna funkcija je assertNotRegex(text, regex, msg=None) koja provjerava da li nema podudaranja između teksta i regularnog izraza.

#### 3.1. Test za funkciju br. 3

Jedna Funkcija: *Contains\_phone\_number*

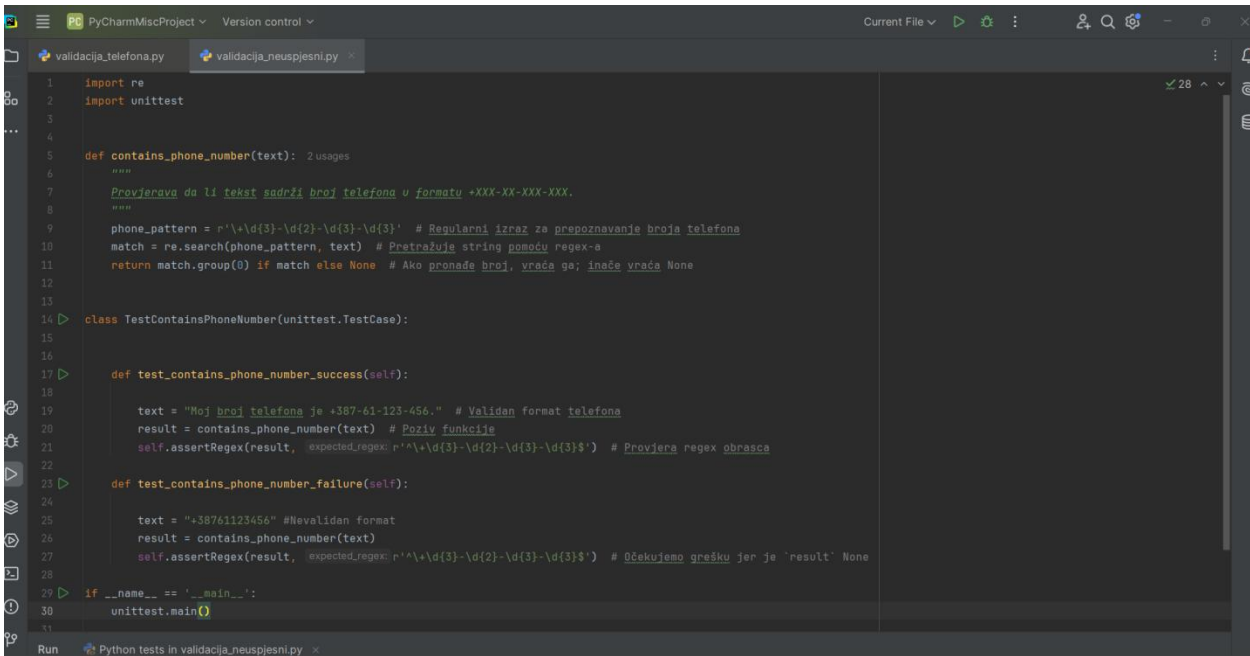
Jedna klasa: *TestContainsPhoneNumber*

Dvije Metode: *test\_contains\_phone\_number\_succes* i *test\_contains\_phone\_number\_failure*

*Opis funkcije:*

Funkcija *Contains\_phone\_number* koristi regularni izraz (r'\+\d{3}-\d{2}-\d{3}-\d{3}') da prepozna broj telefona u formatu +XXX-XX-XXX-XXX gdje \+ označava znak plus, \d{3} označava pozitivni broj zemlje npr. 387, \d{2} označava sljedeće dvije cifre i \d{3} sljedeće 3 cifre kao npr -> +387-62-123-456. Ova funkcija koristi re.search () da pretraži tekst za podudaranje sa gore navedenim obrascem.

Klasa *TestContainsPhoneNumber* nasljeđuje unittest.TestCase.

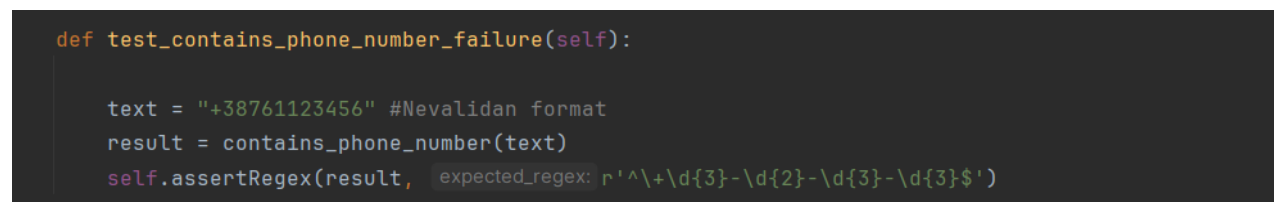


Slika 11 Primjer funkcije br. 3

### Primjer neuspješnog testa:

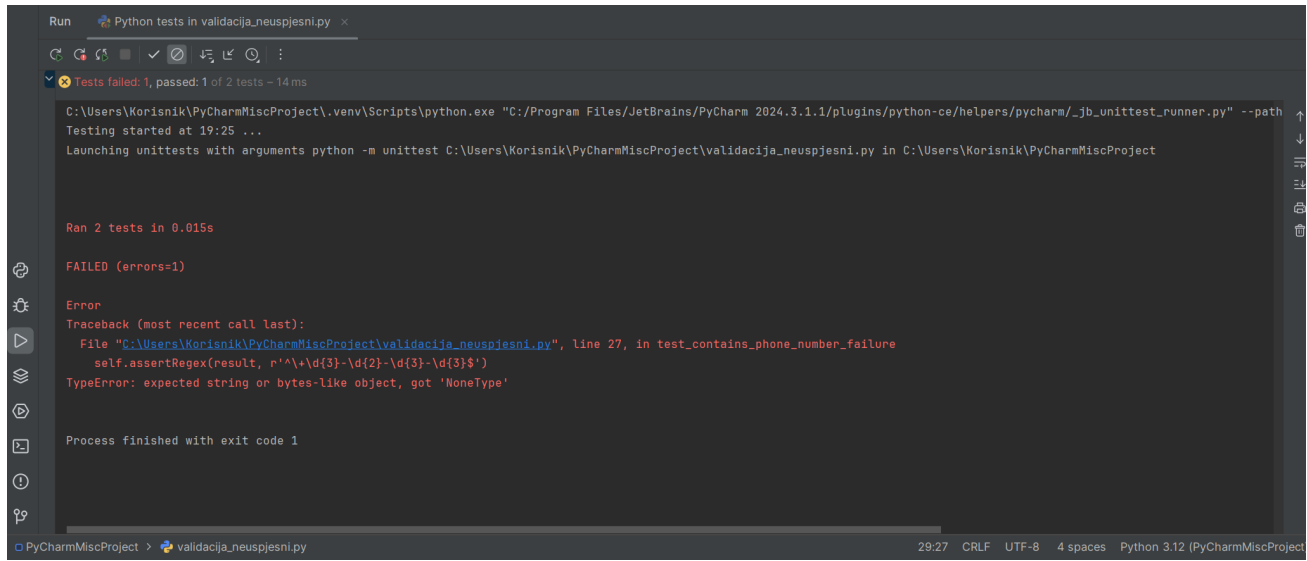
#### OPIS:

Metoda `test_contains_phone_number_failure` testira neuspješan slučaj, gdje broj telefona u tekstu nije u ispravnom formatu, tj. nije u formatu koji je zadan „+XXX-XX-XXX-XXX”.



Slika 12 Primjer metode za neuspješni test za funkciju br. 3

## Rezultat testa:



Slika 13 Rezultat neuspješnog testa za funkciju br. 3

## Primjer uspješnog testa:

### OPIS:

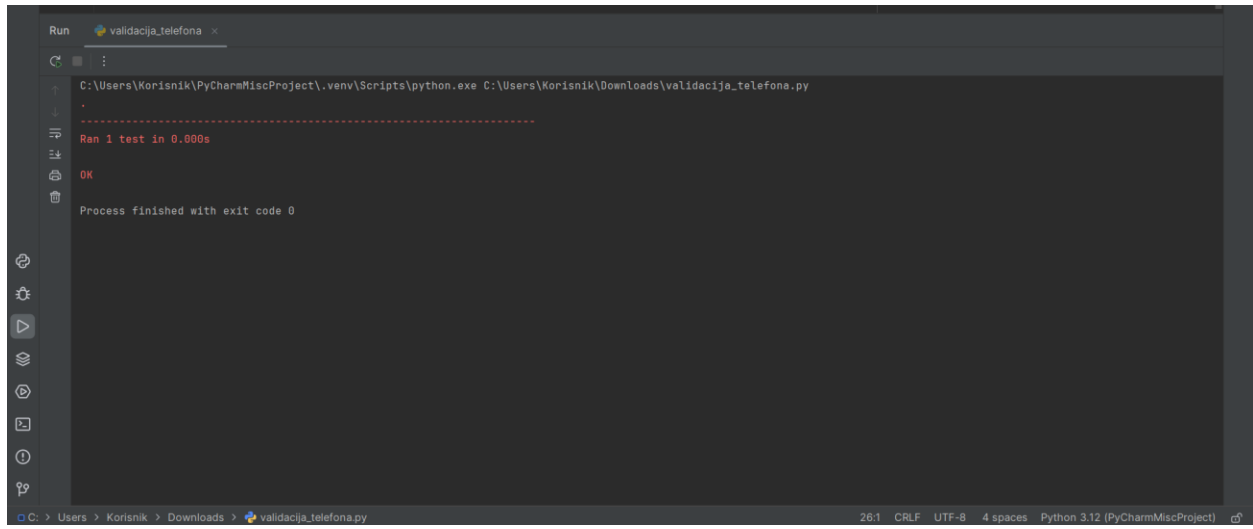
Metoda `test_contains_phone_number_success` kao ulaz prima tekst "Moj broj telefona je +387-61-123-456" koji sadrži validan broj telefona, zatim funkcija `contains_phone_number(text)` poziva se sa tekstom koji treba analizirati, te provjerom da li je rezultat tj. broj telefona u ispravnom formatu. Ako broj telefona odgovara ispravnom obrascu, test će biti uspješan.

```
class TestContainsPhoneNumber(unittest.TestCase):

    def test_contains_phone_number_success(self):
        text = "Moj broj telefona je +387-61-123-456." # Validan broj telefona
        result = contains_phone_number(text) # Poziv funkcije
        self.assertRegex(result, r'^+\d{3}-\d{2}-\d{3}-\d{3}$')
```

Slika 14 Primjer metode za uspješni test za funkciju br. 3

## Rezultat testa:



The screenshot shows the Run console of a PyCharm IDE. The console output is as follows:

```
C:\Users\Korisnik\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\Korisnik\Downloads\validacija_telefona.py  
.  
-----  
Ran 1 test in 0.000s  
OK  
Process finished with exit code 0
```

The status bar at the bottom indicates the file path: C:\Users > Korisnik > Downloads > validacija\_telefona.py, and the encoding settings: 26:1 CRLF UTF-8 4 spaces Python 3.12 (PyCharmMiscProject).

Slika 15 Rezultat uspješnog testa za funkciju br. 3

## 4. assertSetEqual(a,b)

### OPIS:

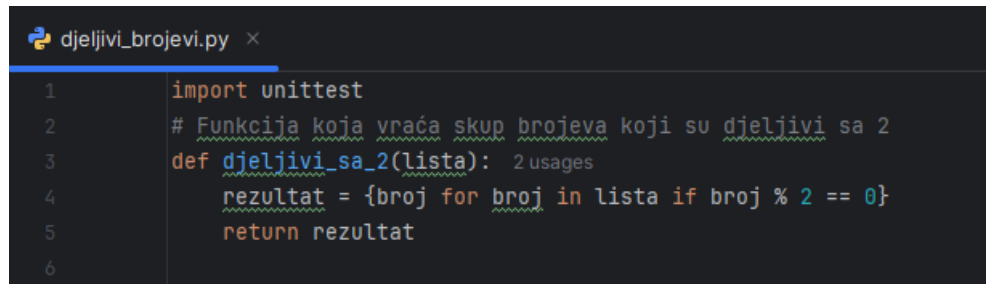
Metoda **assertSetEqual(a,b)** testira da li su dva skupa (set objekta) jednaka, bez obzira na redoslijed elemenata. Odnosno, ova metoda provjerava da li skupovi sadrže iste elemente.

### 4.1. Test za funkciju br. 4

Funkcija br. 4:

Naziv funkcije: *djeljivi\_sa\_2*

Opis funkcije: Funkcija *djeljivi\_sa\_2* provjerava koji su brojevi unutar liste djeljivi sa brojem 2, odnosno provjerava parne brojeve.



```
djeljivi_brojevi.py x
1 import unittest
2 # Funkcija koja vraća skup brojeva koji su djeljivi sa 2
3 def djeljivi_sa_2(lista): 2 usages
4     rezultat = {broj for broj in lista if broj % 2 == 0}
5     return rezultat
6
```

Slika 16: Primjer funkcije br. 4

### Primjer uspješnog testa

### OPIS:

Metoda **test\_uspjesan(self)** testira da li funkcija *djeljivi\_sa\_2* ispravno vraća skup brojeva koji su djeljivi brojem 2 (Slika 17). Lista “brojevi” sadrži testne podatke, odnosno brojeve od 1 do 6. Očekivani rezultat za ovu funkciju je {2, 4, 6}, jer su to brojevi u listi koji zadovoljavaju uslov djeljivosti brojem 2.

***self.assertSetEqual(djeljivi\_sa\_2(brojevi), ocekivani\_skup)*** provjerava da li je rezultat funkcije *djeljivi\_sa\_2* isti kao očekivani skup. S obzirom na to da je uslov funkcije ispunjen, tj. svi brojevi u očekivanom skupu brojeva(*ocekivani\_skup*) djeljivi su brojem 2 i nalaze se u listi *brojevi*, rezultat testa će biti uspješan (Slika 18).

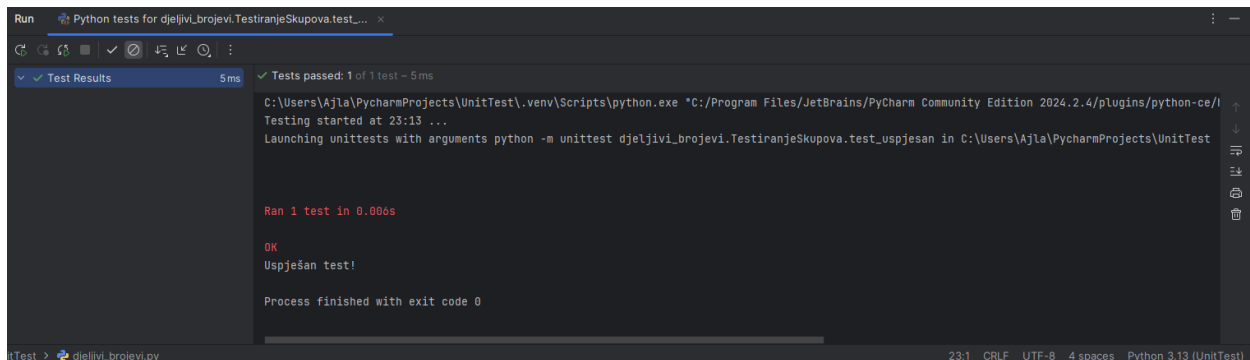
```

11  ▶ def test_uspjesan(self):
12      brojevi = [1, 2, 3, 4, 5, 6]
13      ocekivani_skup = {2, 4, 6}
14      poruka = "Uspješan test!"
15      self.assertEqual(djeljivi_sa_2(brojevi), ocekivani_skup)
16      print(poruka)

```

Slika 17: Primjer metode za uspješni test za funkciju br. 4

### **Rezultat testa:**



The screenshot shows the PyCharm Run window for the test `test_uspjesan`. The output indicates that the test passed successfully. The console output is as follows:

```

C:\Users\Ajla\PycharmProjects\UnitTest\.venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2024.2.4/plugins/python-ce/
Testing started at 23:13 ...
Launching unittests with arguments python -m unittest djeljivi_brojevi.TestiranjeSkupova.test_uspjesan in C:\Users\Ajla\PycharmProjects\UnitTest

Ran 1 test in 0.000s

OK
Uspješan test!

Process finished with exit code 0

```

Slika 18: Rezultat uspješnog testa za funkciju br. 4

### **Primjer neuspješnog testa:**

#### **OPIS:**

Metoda `test_neuspjesan(self)` testira situaciju u kojoj rezultat funkcije nije onaj koji je očekivan (Slika 19). Rezultat testa bit će neuspješan jer skupovi nisu isti, a metoda `assertSetEqual` neće proći ukoliko se skupovi ne poklapaju (Slika 20).

```

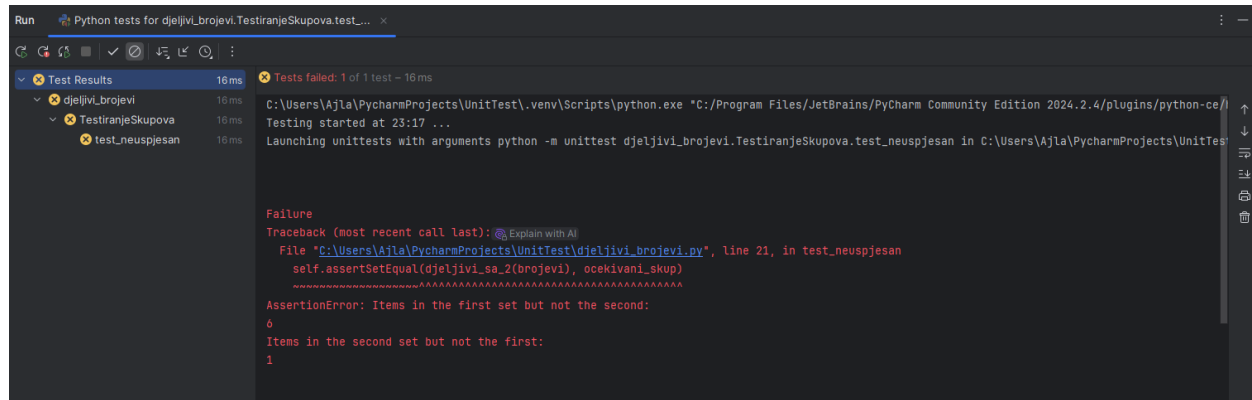
18  ▶ def test_neuspjesan(self):
19      brojevi = [1, 2, 3, 4, 5, 6]
20      ocekivani_skup = {1, 2, 4}
21      self.assertEqual(djeljivi_sa_2(brojevi), ocekivani_skup)
22

```

Slika 19: Primjer metode neuspješnog testa za funkciju br. 4



## Rezultat testa:



Slika 20: Rezultat uspješnog testa za funkciju br. 4

## POPIS PRILOGA

Slika 1 Primjer funkcije br. 1 .....	1
Slika 2 Primjer metode za neuspješni test funkcije br. 1 .....	2
Slika 3 Rezultat neuspješnog testa za funkciju br. 1 .....	2
Slika 4 Primjer metode za uspješni test za funkciju br. 1 .....	3
Slika 5 Rezultat uspješnog testa metode .....	4
Slika 6 Primjer funkcije br. 2 .....	5
Slika 7 Primjer metode za neuspješni test za funkciju br. 2.....	5
Slika 8 Rezultat neuspješnog testa za funkciju br. 2 .....	6
Slika 9 Primjer metode za uspješni test za funkciju br. 2 .....	6
Slika 10 Rezultat uspješnog testa za funkciju br. 2.....	7
Slika 11 Primjer funkcije br. 3 .....	9
Slika 12 Primjer metode za neuspješni test za funkciju br. 3.....	9
Slika 13 Rezultat neuspješnog testa za funkciju br. 3 .....	10
Slika 14 Primjer metode za uspješni test za funkciju br. 3 .....	10
Slika 15 Rezultat uspješnog testa za funkciju br. 3.....	11
Slika 16: Primjer funkcije br. 4.....	12
Slika 17: Primjer metode za uspješni test za funkciju br. 4 .....	13
Slika 18: Rezultat uspješnog testa za funkciju br. 4.....	13
Slika 19: Primjer metode neuspješnog testa za funkciju br. 4 .....	13
Slika 20: Rezultat uspješnog testa za funkciju br. 4.....	14