

## **СОДЕРЖАНИЕ**

|                                    |   |
|------------------------------------|---|
| 1 Лабораторная работа № 9-12 ..... | 2 |
|------------------------------------|---|

# 1 Лабораторная работа № 9-12

Тема: Профилирование кода средствами инструментальной среды разработки. Разработка тестовых модулей проекта. Выполнение функционального тестирования и тестирования интеграции. Документирование результатов. Тестирование интерфейса пользователя.

Цель: использование технологий и инструментов отладки и тестирования, встроенных в Visual Studio для выполнения тестирования разного типа с целью обнаружения ошибок и их дальнейшее исправление, ведение документации и анализ результатов тестирования.

1. Выполните профилирование кода сайта на Bottle из предыдущей ЛР с помощью команды Отладка > Запустить профилирование Python (Debug > Launch python profiling...), по умолчанию в окне настроек должно отобразиться название проекта:

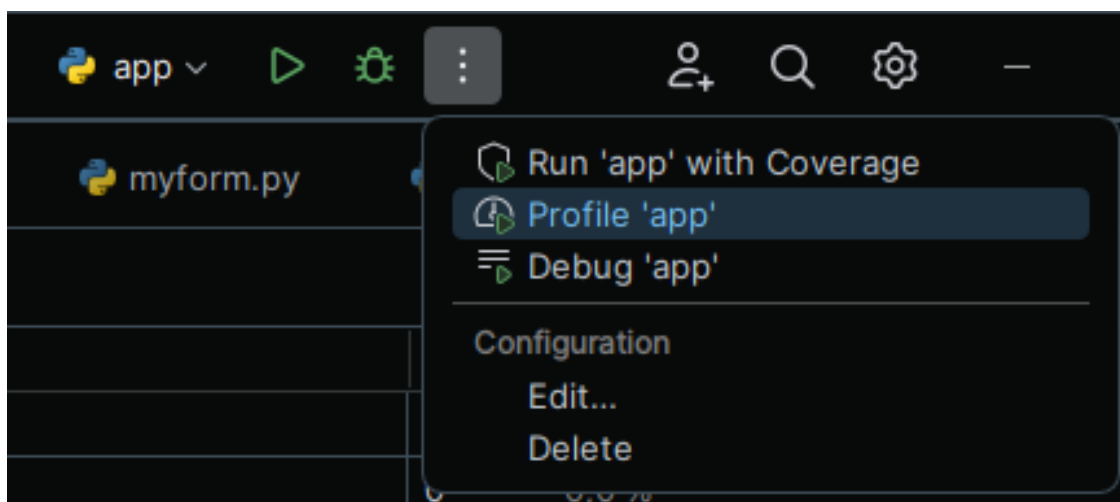


Рисунок 1 – Profiling в PyCharm

Полученный отчёт:

| Name  | Call Count | Time (ms)   | Own Time (ms) |
|---|------------|-------------|---------------|
| <built-in method select.select>             | 7          | 3215 98,6 % | 3215 98,6 %   |
| <built-in method _socket.gethostbyaddr>     | 1          | 6 0,2 %     | 4 0,1 %       |
| <built-in method nt.stat>                   | 190        | 3 0,1 %     | 3 0,1 %       |
| _path_join                                  | 684        | 4 0,1 %     | 2 0,1 %       |
| <built-in method builtins.__build_class__>  | 174        | 7 0,2 %     | 2 0,1 %       |
| <method 'bind' of '_socket.socket' objects> | 1          | 2 0,1 %     | 2 0,1 %       |
| _parse                                      | 71         | 4 0,1 %     | 1 0,0 %       |
| _new_module                                 | 22         | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| acquire                                     | 28         | 0 0,0 %     | 0 0,0 %       |
| release                                     | 28         | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| __enter__                                   | 28         | 0 0,0 %     | 0 0,0 %       |
| __exit__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| cb  | 28         | 0 0,0 %     | 0 0,0 %       |
| _get_module_lock                            | 28         | 0 0,0 %     | 0 0,0 %       |
| _verbose_message                            | 702        | 0 0,0 %     | 0 0,0 %       |
| _requires_builtin_wrapper                   | 1          | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 26         | 0 0,0 %     | 0 0,0 %       |
| cached                                      | 45         | 0 0,0 %     | 0 0,0 %       |

Рисунок 2 – Отчёт профилирования

Сравнение двух отчётов:

| Name  | Call Count | Time (ms)   | Own Time (ms) |
|---|------------|-------------|---------------|
| <built-in method select.select>             | 7          | 3215 98,6 % | 3215 98,6 %   |
| <built-in method _socket.gethostbyaddr>     | 1          | 6 0,2 %     | 4 0,1 %       |
| <built-in method nt.stat>                   | 190        | 3 0,1 %     | 3 0,1 %       |
| _path_join                                  | 684        | 4 0,1 %     | 2 0,1 %       |
| <built-in method builtins.__build_class__>  | 174        | 7 0,2 %     | 2 0,1 %       |
| <method 'bind' of '_socket.socket' objects> | 1          | 2 0,1 %     | 2 0,1 %       |
| _parse                                      | 71         | 4 0,1 %     | 1 0,0 %       |
| _new_module                                 | 22         | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| acquire                                     | 28         | 0 0,0 %     | 0 0,0 %       |
| release                                     | 28         | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| __enter__                                   | 28         | 0 0,0 %     | 0 0,0 %       |
| __exit__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| cb  | 28         | 0 0,0 %     | 0 0,0 %       |
| _get_module_lock                            | 28         | 0 0,0 %     | 0 0,0 %       |
| _verbose_message                            | 702        | 0 0,0 %     | 0 0,0 %       |
| _requires_builtin_wrapper                   | 1          | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 26         | 0 0,0 %     | 0 0,0 %       |
| cached                                      | 45         | 0 0,0 %     | 0 0,0 %       |

| Name  | Call Count | Time (ms)   | Own Time (ms) |
|---|------------|-------------|---------------|
| <built-in method select.select>             | 190        | 3215 98,6 % | 9576 39,5 %   |
| <method 'recv' of '_socket.socket' objects> | 2          | 107 3,3 %   | 107 3,3 %     |
| <built-in method nt.stat>                   | 194        | 4 0,1 %     | 4 0,1 %       |
| <built-in method _socket.gethostbyaddr>     | 1          | 6 0,2 %     | 4 0,1 %       |
| _path_join                                  | 684        | 4 0,1 %     | 2 0,1 %       |
| select                                      | 190        | 3215 98,6 % | 2 0,1 %       |
| <built-in method builtins.__build_class__>  | 174        | 7 0,2 %     | 2 0,1 %       |
| <method 'bind' of '_socket.socket' objects> | 1          | 2 0,1 %     | 2 0,1 %       |
| _parse                                      | 71         | 4 0,1 %     | 1 0,0 %       |
| _select                                     | 190        | 3215 98,6 % | 1 0,0 %       |
| _new_module                                 | 22         | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| acquire                                     | 28         | 0 0,0 %     | 0 0,0 %       |
| release                                     | 28         | 0 0,0 %     | 0 0,0 %       |
| __init__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| __enter__                                   | 28         | 0 0,0 %     | 0 0,0 %       |
| __exit__                                    | 28         | 0 0,0 %     | 0 0,0 %       |
| cb  | 28         | 0 0,0 %     | 0 0,0 %       |
| _get_module_lock                            | 28         | 0 0,0 %     | 0 0,0 %       |
| _verbose_message                            | 702        | 0 0,0 %     | 0 0,0 %       |

Рисунок 3 – Сравнение двух отчётов

2. Выполните примеры unit-тестов для простейшего модуля Python с функциями калькулятора. Создайте консольное приложение как в ЛР№1. Модуль calc.py будет представлять собой библиотеку, содержащую функции для выполнения основных арифметических действий:

```
def add(a, b):
    return a + b
```

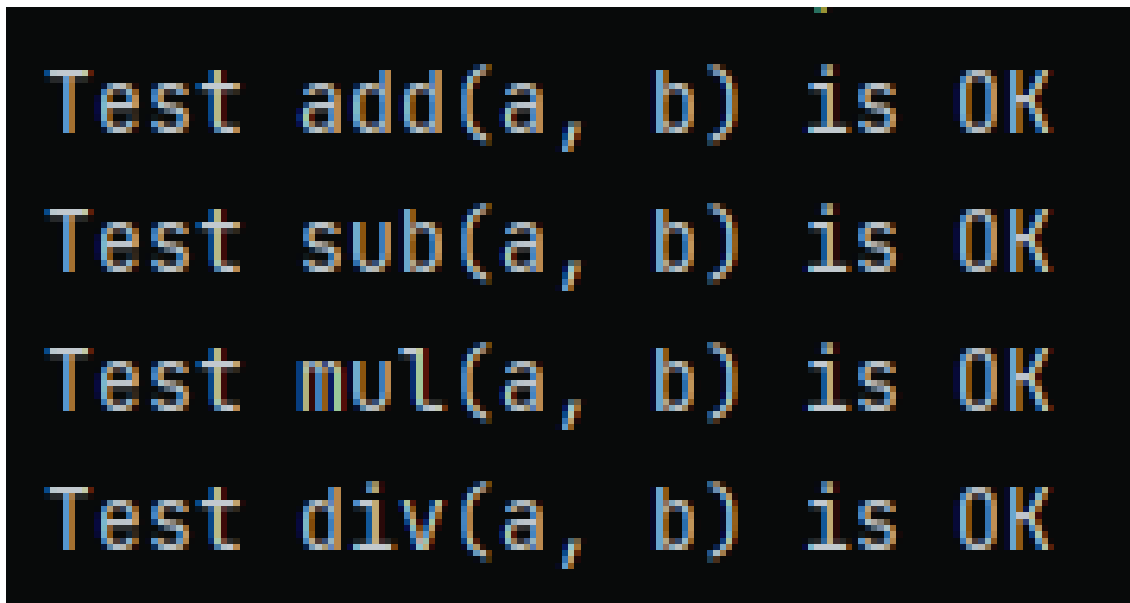
```
def sub(a, b):  
    return a - b  
  
def mul(a, b):  
    return a * b  
  
def div(a, b):  
    return a / b
```

Для того, чтобы протестировать эту библиотеку, создайте отдельный файл с названием `testcalc.py` и поместите туда функции, которые проверяют корректность работы модуля:

```
import calc  
  
def test_add():  
    if calc.add(1, 2) == 3:  
        print("Test add(a, b) is OK")  
    else:  
        print("Test add(a, b) is Fail")  
  
def test_sub():  
    if calc.sub(4, 2) == 2:  
        print("Test sub(a, b) is OK")  
    else:  
        print("Test sub(a, b) is Fail")  
  
def test_mul():  
    if calc.mul(2, 5) == 10:  
        print("Test mul(a, b) is OK")  
    else:  
        print("Test mul(a, b) is Fail")  
  
def test_div():  
    if calc.div(8, 4) == 2:  
        print("Test div(a, b) is OK")  
    else:  
        print("Test div(a, b) is Fail")
```

```
test_add()
test_sub()
test_mul()
test_div()
```

Запустите `test_calc.py`, выбрав в его контекстном меню Запуск без отладки.



```
Test add(a, b) is OK
Test sub(a, b) is OK
Test mul(a, b) is OK
Test div(a, b) is OK
```

Рисунок 4 – Тестирование `calc`

3. Для тестирования набора функций из `calc.py` с помощью `unittest` создайте файл с именем `utest_calc.py` и следующим кодом:

```
import calc

def test_add():
    if calc.add(1, 2) == 3:
        print("Test add(a, b) is OK")
    else:
        print("Test add(a, b) is Fail")

def test_sub():
    if calc.sub(4, 2) == 2:
        print("Test sub(a, b) is OK")
    else:
        print("Test sub(a, b) is Fail")
```

```
def test_mul():
    if calc.mul(2, 5) == 10:
        print("Test mul(a, b) is OK")
    else:
        print("Test mul(a, b) is Fail")

def test_div():
    if calc.div(8, 4) == 2:
        print("Test div(a, b) is OK")
    else:
        print("Test div(a, b) is Fail")

test_add()
test_sub()
test_mul()
test_div()
```

```
(venv) PS C:\Users\user\PycharmProjects\testing> python .\utest_calc.py
....
-----
Ran 4 tests in 0.000s
OK
```

Рисунок 5 – Запуск тестов

Можно сделать запрос расширенной информации по пройденным тестам, для этого необходимо добавить ключ `-v`. Запустите скрипт `utestcalc.py` в терминале посредством контекстного меню. Введите команду `python -m unittest -v utestcalc.py`.

```
(venv) PS C:\Users\user\PycharmProjects\testing> python -m unittest -v utest_calc.py
test_add (utest_calc.CalcTest) ... ok
test_div (utest_calc.CalcTest) ... ok
test_mul (utest_calc.CalcTest) ... ok
test_sub (utest_calc.CalcTest) ... ok
-----
Ran 4 tests in 0.000s
```

Рисунок 6 – Показ всех тестов

4. Создайте ещё одно консольное приложение. Подготовьте VS для создания unit-тестов с помощью фреймворка. Для этого щелкните проект правой кнопкой мыши в обозревателе решений и выберите платформу unittest на вкладке Тест в области свойств (или нажав на значок свойств в панели его инструментов).

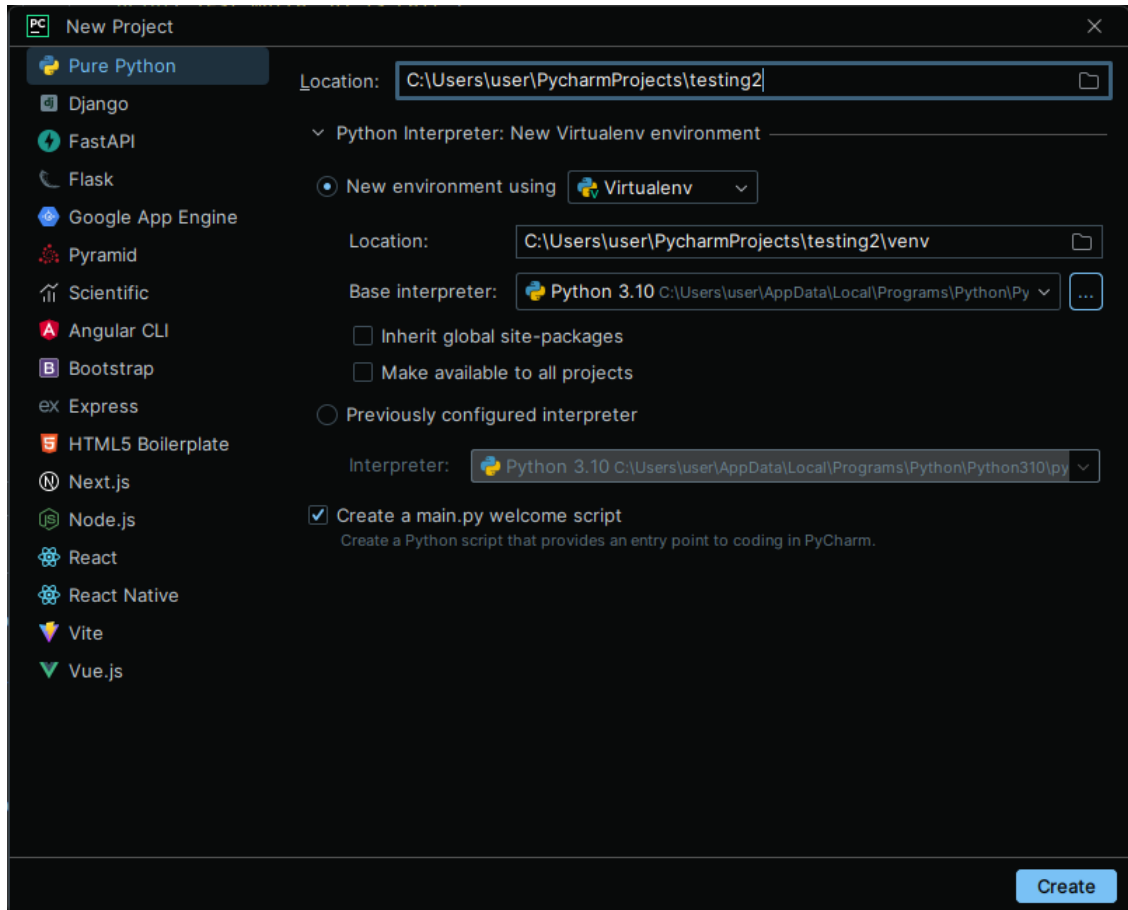


Рисунок 7 – Создание проекта в PyCharm

Создайте два unit-теста для проверки работы регулярного выражения на соответствие email (вынесите формирование и сравнение со строкой в главный модуль `myform_mail.py`), в одном используя метод `assertFalse`, а в другом – `assertTrue` (подробнее о методах в официальной документации или статье <https://tirinox.ru/unit-test-python/>).

В unit-тесте с `assertFalse` в цикле проверяйте список с примерами email неверного формата (перечислить возможные тестовые случаи, не менее 12 с разными типами ошибок), например, `list_mailuncor = [`,  
`”1 ”m1@ ”@mail ...]`, а в unit-тесте с `assertTrue` – список с корректны-

ми email, к примеру, `list_mailcor = ["m.m@mail.ru" "m1@gmail.com ..."]`.

```
from re import fullmatch
```

```
def mail_is_correct(mail: str):
```

Код 1: my\_form\_mail.py

```
return
```

```
↪ fullmatch(r"[a-zA-Z0-9._]{1,256}@[a-zA-Z0-9]{1,100}\.[a-zA-Z0-9]{1,7}",
```

```
↪ mail) is not None
```

```
\
```



```

import unittest
from my_form_mail import *

class FormTest(unittest.TestCase):
    list_mail_uncor = ["", "a@", "pa", "pank@a", "@a", "pank@pank", "
↪ @ . ", "pank@pank.", "pank@pank,su",
                        "pank@pank.su, pank@pank.su", "
↪ "vasya.pankov@ma", "pa@pa@.com",
↪ "misha...palok"]
    list_mail_cor = ["pank@pank.su", "vasya.pankov26@gamil.com",
↪ "vasya_@mail.com",
↪ "veryveryveryveryveryveryveryveryveryveryveryveryveryveryvery@longlo
↪ "&=_'+@sym.love", "SENSETIVE@UPPER.CASE"]

    def test_F_mail(self):
        for mail in self.list_mail_uncor:
            self.assertFalse(mail_is_correct(mail))

    def test_T_mail(self):
        for mail in self.list_mail_cor:
            self.assertTrue(mail_is_correct(mail))

```

Код 2: ctest\_form\_mail.py

Ошибка:

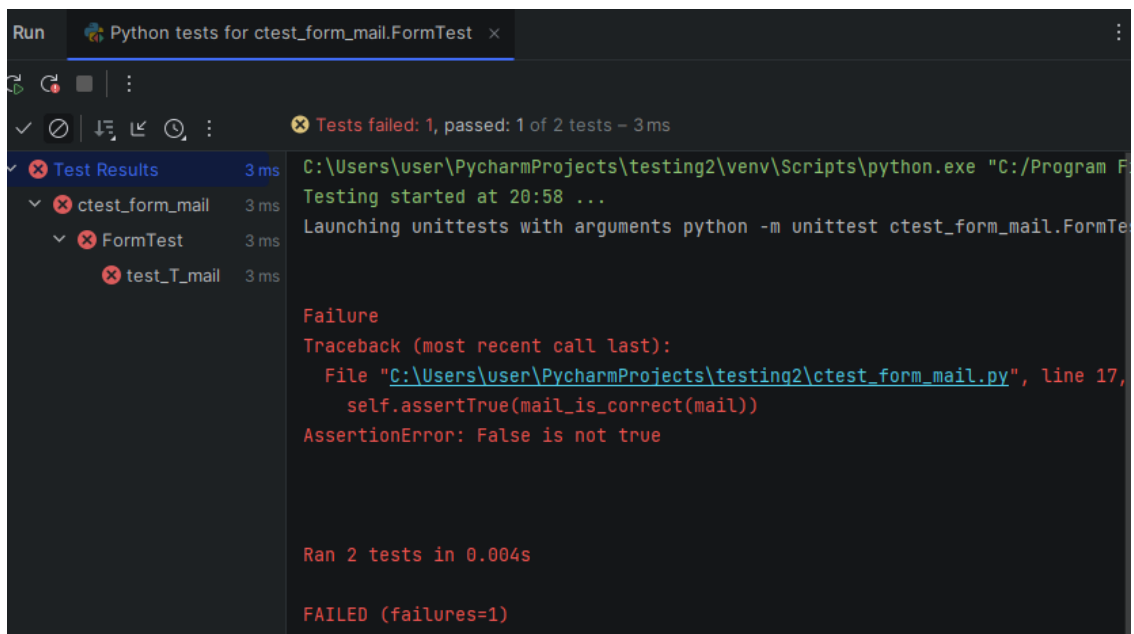


Рисунок 8 – Демонстрация ошибки

Суть ошибки в том, что все не все символы были в проверке, которые могут быть в почте могут быть в почте.

Исправленная версия регулярного выражения:

```
r"[a-zA-Z0-9._&=\\-+]{1,256}@[a-zA-Z0-9]{1,100}\\.[a-zA-Z0-9]{1,256}"
```

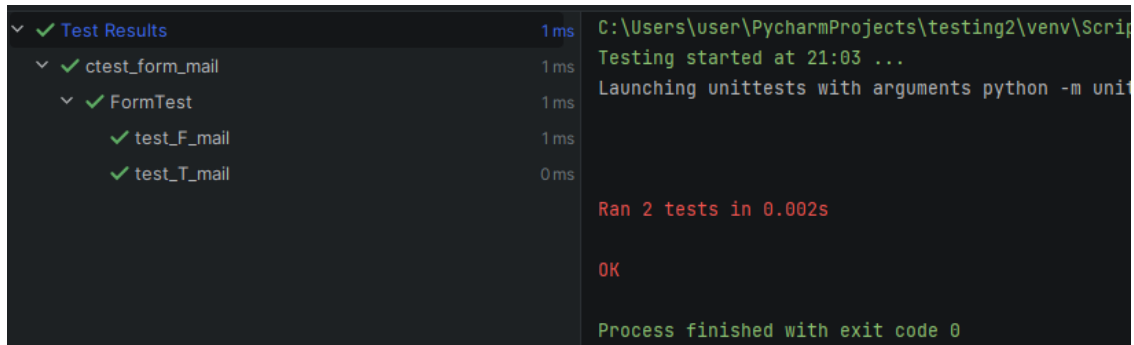


Рисунок 9 – Исправленная версия

5. Вернитесь к приложению Bottle. Интегрируйте модули кода с проверкой работы регулярного выражения и соответствующими unit-тестами в свой проект. Выполните запуск тестов и отладку работы проекта. Зафиксируйте в Git соответствующие изменения (сделайте не менее 2 коммитов). Сделайте push на Github. Задокументируйте все результаты работы.