

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

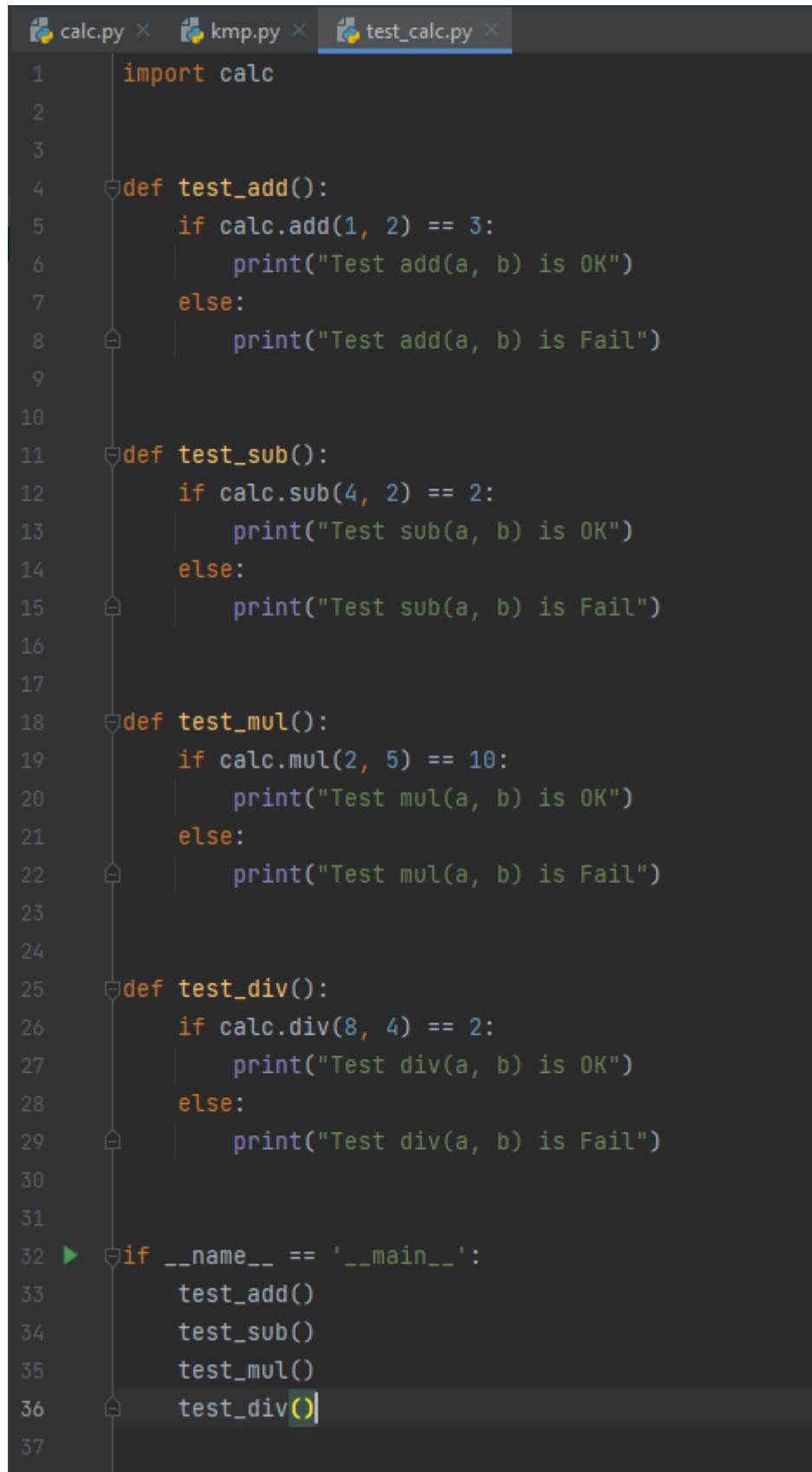
**Отчет о лабораторной работе №10 по дисциплине основы программной
инженерии**

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

1. Примеры из методических указаний



```
1  import calc
2
3
4  def test_add():
5      if calc.add(1, 2) == 3:
6          print("Test add(a, b) is OK")
7      else:
8          print("Test add(a, b) is Fail")
9
10
11 def test_sub():
12     if calc.sub(4, 2) == 2:
13         print("Test sub(a, b) is OK")
14     else:
15         print("Test sub(a, b) is Fail")
16
17
18 def test_mul():
19     if calc.mul(2, 5) == 10:
20         print("Test mul(a, b) is OK")
21     else:
22         print("Test mul(a, b) is Fail")
23
24
25 def test_div():
26     if calc.div(8, 4) == 2:
27         print("Test div(a, b) is OK")
28     else:
29         print("Test div(a, b) is Fail")
30
31
32 if __name__ == '__main__':
33     test_add()
34     test_sub()
35     test_mul()
36     test_div()
37
```

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

Process finished with exit code 0
```

```
1 ▶  #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4
5    import unittest
6    import calc
7
8
9 ▶   class CalcTest(unittest.TestCase):
10 ▶       def test_add(self):
11         self.assertEqual(calc.add(1, 2), 3)
12
13 ▶       def test_sub(self):
14         self.assertEqual(calc.sub(4, 2), 2)
15
16 ▶       def test_mul(self):
17         self.assertEqual(calc.mul(2, 5), 10)
18
19 ▶       def test_div(self):
20         self.assertEqual(calc.div(8, 4), 2)
21
22
23 ▶   if __name__ == '__main__':
24       unittest.main()
25
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python -m unittest utest_calc.py
....
-----
Ran 4 tests in 0.001s

OK
PS C:\Users\student-09-525\PycharmProjects\kmp\examples>
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> 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 0.004s

OK
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> 
```

```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5  ▶  import unittest
6  ▶  import calc
7
8
9  ▶  class CalcTest(unittest.TestCase):
10     """Calc tests"""
11
12     @classmethod
13     ▶ def setUpClass(cls):
14         """Set up for class"""
15         print("setUpClass")
16         print("=====")
17
18     @classmethod
19     ▶ def tearDownClass(cls):
20         """Tear down for class"""
21         print("=====")
22         print("tearDownClass")
23
24     ▶ def setUp(self):
25         """Set up for test"""
26         print("Set up for [" + self.shortDescription() + "]")
27
28     ▶ def tearDown(self):
29         """Tear down for test"""
30         print("Tear down for [" + self.shortDescription() + "]")
31         print("")
32
33     ▶ def test_add(self):
34         """Add operation test"""
35         print("id: " + self.id())
36         self.assertEqual(calc.add(1, 2), 3)
37
38     ▶ def test_sub(self):
39         """Sub operation test"""
40         print("id: " + self.id())
41         self.assertEqual(calc.sub(4, 2), 2)
42
43     ▶ def test_mul(self):
44         """Mul operation test"""
45         print("id: " + self.id())
46         self.assertEqual(calc.mul(2, 5), 10)
47
48     ▶ def test_div(self):
49         """Div operation test"""
50         print("id: " + self.id())
51         self.assertEqual(calc.div(8, 4), 2)
52
53
54 ▶  if __name__ == '__main__':
55     unittest.main()

```

CalcTest

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python -m unittest -v utest_calc.py
setUpClass
=====
test_add (utest_calc.CalcTest)
Add operation test ... Set up for [Add operation test]
id: utest_calc.CalcTest.test_add
Tear down for [Add operation test]

ok
test_div (utest_calc.CalcTest)
Div operation test ... Set up for [Div operation test]
id: utest_calc.CalcTest.test_div
Tear down for [Div operation test]

ok
test_mul (utest_calc.CalcTest)
Mul operation test ... Set up for [Mul operation test]
id: utest_calc.CalcTest.test_mul
Tear down for [Mul operation test]

ok
test_sub (utest_calc.CalcTest)
Sub operation test ... Set up for [Sub operation test]
id: utest_calc.CalcTest.test_sub
Tear down for [Sub operation test]

ok
=====
tearDownClass

-----
Ran 4 tests in 0.006s

OK
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> 
```


```
utest_calc.py × calc_tests.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 import unittest
6 import calc
7
8
9 class CalcTest(unittest.TestCase):
10     def test_add(self):
11         self.assertEqual(calc.add(1, 2), 3)
12
13     def test_sub(self):
14         self.assertEqual(calc.sub(4, 2), 2)
15
16     def test_mul(self):
17         self.assertEqual(calc.mul(2, 5), 10)
18
19     def test_div(self):
20         self.assertEqual(calc.div(8, 4), 2)
21
```

```
utest_calc.py × calc_tests.py × test_runner.py ×
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4
5 import unittest
6 import calc_tests
7
8
9 calcTestSuite = unittest.TestSuite()
10 calcTestSuite.addTest(unittest.makeSuite(calc_tests.CalcTest))
11
12 runner = unittest.TextTestRunner(verbosity=2)
13 runner.run(calcTestSuite)
14
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python test_runner.py
test_add (calc_tests.CalcTest) ... ok
test_div (calc_tests.CalcTest) ... ok
test_mul (calc_tests.CalcTest) ... ok
test_sub (calc_tests.CalcTest) ... ok
```

Ran 4 tests in 0.002s

OK



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import unittest
6  import calc
7
8
9  class CalcBasicTests(unittest.TestCase):
10     def test_add(self):
11         self.assertEqual(calc.add(1, 2), 3)
12
13     def test_sub(self):
14         self.assertEqual(calc.sub(4, 2), 2)
15
16     def test_mul(self):
17         self.assertEqual(calc.mul(2, 5), 10)
18
19     def test_div(self):
20         self.assertEqual(calc.div(8, 4), 2)
21
22
23  class CalcExTests(unittest.TestCase):
24     def test_sqrt(self):
25         self.assertEqual(calc.sqrt(4), 2)
26
27     def test_pow(self):
28         self.assertEqual(calc.pow(3, 3), 27)
29
```



```
utest_calc.py × calc_tests.py × test_runner.py ×
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      import unittest
6      import calc_tests
7
8
9      calcTestSuite = unittest.TestSuite()
10     calcTestSuite.addTest(unittest.makeSuite(calc_tests.CalcBasicTests))
11     calcTestSuite.addTest(unittest.makeSuite(calc_tests.CalcExTests))
12     print("count of tests: " + str(calcTestSuite.countTestCases()) + "\n")
13
14     runner = unittest.TextTestRunner(verbosity=2)
15     runner.run(calcTestSuite)
16
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python test_runner.py
count of tests: 6
```

```
test_add (calc_tests.CalcBasicTests) ... ok
test_div (calc_tests.CalcBasicTests) ... ok
test_mul (calc_tests.CalcBasicTests) ... ok
test_sub (calc_tests.CalcBasicTests) ... ok
test_pow (calc_tests.CalcExTests) ... ok
test_sqrt (calc_tests.CalcExTests) ... ok
```

```
-----
Ran 6 tests in 0.002s
```

```
OK
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> █
```

```
utest_calc.py × calc_tests.py × test_runner.py ×
1  ▶ 1 #!/usr/bin/env python3
2    2 # -*- coding: utf-8 -*-
3
4
5    5 import unittest
6    6 import calc_tests
7
8
9    9 testCases = [calc_tests.CalcBasicTests, calc_tests.CalcExTests]
10   10 testLoad = unittest.TestLoader()
11   11 suites = []
12   12 for tc in testCases:
13   13     suites.append(testLoad.loadTestsFromTestCase(tc))
14
15   15 res_suite = unittest.TestSuite(suites)
16   16 runner = unittest.TextTestRunner(verbosity=2)
17   17 runner.run(res_suite)
18
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python test_runner.py
test_add (calc_tests.CalcBasicTests) ... ok
test_div (calc_tests.CalcBasicTests) ... ok
test_mul (calc_tests.CalcBasicTests) ... ok
test_sub (calc_tests.CalcBasicTests) ... ok
test_pow (calc_tests.CalcExTests) ... ok
test_sqrt (calc_tests.CalcExTests) ... ok

-----
Ran 6 tests in 0.005s

OK
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> █
```

```
utest_calc.py × calc_tests.py × test_runner.py ×
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      import unittest
6      import calc_tests
7
8
9      testLoad = unittest.TestLoader()
10     suites = testLoad.loadTestsFromModule(calc_tests)
11     runner = unittest.TextTestRunner(verbosity=2)
12     runner.run(suites)
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python test_runner.py
test_add (calc_tests.CalcBasicTests) ... ok
test_div (calc_tests.CalcBasicTests) ... ok
test_mul (calc_tests.CalcBasicTests) ... ok
test_sub (calc_tests.CalcBasicTests) ... ok
test_pow (calc_tests.CalcExTests) ... ok
test_sqrt (calc_tests.CalcExTests) ... ok

-----
Ran 6 tests in 0.002s

OK
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> 
```

```
utest_calc.py × calc_tests.py × test_runner.py ×
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import unittest
6  import calc_tests
7
8
9  testLoad = unittest.TestLoader()
10 suites = testLoad.loadTestsFromModule(calc_tests)
11 testResult = unittest.TestResult()
12 runner = unittest.TextTestRunner(verbosity=1)
13 testResult = runner.run(suites)
14 print("errors")
15 print(len(testResult.errors))
16 print("failures")
17 print(len(testResult.failures))
18 print("skipped")
19 print(len(testResult.skipped))
20 print("testsRun")
21 print(testResult.testsRun)
22 |
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python test_runner.py
.....
-----
Ran 6 tests in 0.001s

OK
errors
0
failures
0
skipped
0
testsRun
6
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> |
```

```
utest_calc.py × calc_tests.py × test_runner.py ×
1 ▶ 1 #!/usr/bin/env python3
2 2 # -*- coding: utf-8 -*-
3
4
5 5 import unittest
6 6 import calc
7
8
9 ▶ 9 class CalcBasicTests(unittest.TestCase):
10 10     @unittest.skip("Temporary skip test_add")
11 ▶ 11     def test_add(self):
12 12         self.assertEqual(calc.add(1, 2), 3)
13
14 ▶ 14     def test_sub(self):
15 15         self.assertEqual(calc.sub(4, 2), 2)
16
17 ▶ 17     def test_mul(self):
18 18         self.assertEqual(calc.mul(2, 5), 10)
19
20 ▶ 20     def test_div(self):
21 21         self.assertEqual(calc.div(8, 4), 2)
22
23
24 ▶ 24 class CalcExTests(unittest.TestCase):
25 ▶ 25     def test_sqrt(self):
26 26         self.assertEqual(calc.sqrt(4), 2)
27
28 ▶ 28     def test_pow(self):
29 29         self.assertEqual(calc.pow(3, 3), 27)
30
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python test_runner.py
S.....
-----
Ran 6 tests in 0.002s

OK (skipped=1)
errors
0
failures
0
skipped
1
testsRun
6
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> 
```

```
utest_calc.py × calc_tests.py × test_runner.py ×
1 ▶ 1 #!/usr/bin/env python3
2 2 # -*- coding: utf-8 -*-
3
4
5 5 import unittest
6 6 import calc
7
8
9 ▶ 9 class CalcBasicTests(unittest.TestCase):
10 ▶ 10     def test_add(self):
11 11         self.assertEqual(calc.add(1, 2), 3)
12
13 ▶ 13     def test_sub(self):
14 14         self.assertEqual(calc.sub(4, 2), 2)
15
16 ▶ 16     def test_mul(self):
17 17         self.assertEqual(calc.mul(2, 5), 10)
18
19 ▶ 19     def test_div(self):
20 20         self.assertEqual(calc.div(8, 4), 2)
21
22
23 23 @unittest.skip("Skip CalcExTests")
24 ▶ 24 class CalcExTests(unittest.TestCase):
25 ▶ 25     def test_sqrt(self):
26 26         self.assertEqual(calc.sqrt(4), 2)
27
28 ▶ 28     def test_pow(self):
29 29         self.assertEqual(calc.pow(3, 3), 27)
30
```

```
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> python test_runner.py
....SS
-----
Ran 6 tests in 0.001s

OK (skipped=2)
errors
0
failures
0
skipped
2
testsRun
6
PS C:\Users\student-09-525\PycharmProjects\kmp\examples> 
```

2. Индивидуальное задание (рис. 1-15).


```
flights.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      import argparse
6          import sqlite3
7          import typing as t
8      from pathlib import Path
9
10
11      def display_flights(flights: t.List[t.Dict[str, t.Any]]) -> None:
12          """
13              Отобразить список рейсов.
14          """
15          if flights:
16              # Заголовок таблицы.
17              line = '+--{}--{}--{}--{}--+'.format(
18                  '-' * 4,
19                  '-' * 30,
20                  '-' * 20,
21                  '-' * 15
22              )
23              print(line)
24              print(
25                  '| {:^4} | {:^30} | {:^20} | {:^15} |'.format(
26                      "No",
27                      "Пункт назначения",
28                      "Номер рейса",
29                      "Тип самолёта"
30                  )
31              )
32              print(line)
33              for idx, flight in enumerate(flights, 1):
34                  print(
35                      '| {:>4} | {:<30} | {:<20} | {:<15} |'.format(
36                          idx,
37                          flight.get('flight_destination', ''),
38                          flight.get('flight_number', ''),
39                          flight.get('airplane_type', 0)
```

Рисунок 1 – Код программы

```

40         )
41     )
42     print(line)
43     else:
44         print("Список рейсов пуст.")
45
46
47 def create_db(database_path: Path) -> None:
48     """
49     Создать базу данных.
50     """
51     conn = sqlite3.connect(database_path)
52     cursor = conn.cursor()
53     cursor.execute(
54         """
55         CREATE TABLE IF NOT EXISTS flight_numbers (
56             num_id INTEGER PRIMARY KEY AUTOINCREMENT,
57             num_title TEXT NOT NULL
58         )
59         """
60     )
61     cursor.execute(
62         """
63         CREATE TABLE IF NOT EXISTS flights (
64             flight_id INTEGER PRIMARY KEY AUTOINCREMENT,
65             flight_destination TEXT NOT NULL,
66             num_id INTEGER NOT NULL,
67             airplane_type TEXT NOT NULL,
68             FOREIGN KEY(num_id) REFERENCES flight_numbers(num_id)
69         )
70         """
71     )
72     conn.close()
73
74
75 def add_flight(
76     database_path: Path,
77     flight_destination: str,
78     flight_number: str,

```

Рисунок 2 – Код программы, продолжение

```

79         airplane_type: str
80     ) -> None:
81         """
82         Добавить рейс в базу данных.
83         """
84         conn = sqlite3.connect(database_path)
85         cursor = conn.cursor()
86         cursor.execute(
87             """
88             SELECT num_id FROM flight_numbers WHERE num_title = ?
89             """,
90             (flight_number,)
91         )
92         row = cursor.fetchone()
93         if row is None:
94             cursor.execute(
95                 """
96                 INSERT INTO flight_numbers (num_title) VALUES (?)
97                 """,
98                 (flight_number,)
99             )
100             num_id = cursor.lastrowid
101         else:
102             num_id = row[0]
103         cursor.execute(
104             """
105             INSERT INTO flights (flight_destination, num_id, airplane_type)
106             VALUES (?, ?, ?)
107             """,
108             (flight_destination, num_id, airplane_type)
109         )
110         conn.commit()
111         conn.close()
112
113
114     def select_all(database_path: Path) -> t.List[t.Dict[str, t.Any]]:
115         """
116         Выбрать все рейсы.
117         """

```

Рисунок 3 – Код программы, продолжение

```

118     conn = sqlite3.connect(database_path)
119     print(database_path)
120     cursor = conn.cursor()
121     cursor.execute(
122         """SELECT flights.flight_destination, flight_numbers.num_title,
123             flights.airplane_type FROM flights INNER JOIN flight_numbers ON
124             flight_numbers.num_id = flights.num_id """
125     )
126     rows = cursor.fetchall()
127     conn.close()
128     ret = [
129         {
130             "flight_destination": row[0],
131             "flight_number": row[1],
132             "airplane_type": row[2],
133         }
134         for row in rows
135     ]
136     return ret
137
138
139     def select_flights(
140         database_path: Path, air_type: str
141     ) -> t.List[t.Dict[str, t.Any]]:
142         """
143         Выбрать все рейсы заданного типа.
144         """
145         conn = sqlite3.connect(database_path)
146         cursor = conn.cursor()
147         cursor.execute(
148             """
149             SELECT flights.flight_destination, flight_numbers.num_title,
150             flights.airplane_type
151             FROM flights
152             INNER JOIN flight_numbers ON flight_numbers.num_id = flights.num_id
153             WHERE flights.airplane_type == ?
154             """
155             ,
156             (air_type,)

```

Рисунок 4 – Код программы, продолжение

```

157     rows = cursor.fetchall()
158     conn.close()
159     ret = [
160         {
161             "flight_destination": row[0],
162             "flight_number": row[1],
163             "airplane_type": row[2],
164         }
165         for row in rows
166     ]
167     return ret
168
169
170 def main(command_line=None):
171     # Создать родительский парсер для определения имени файла.
172     file_parser = argparse.ArgumentParser(add_help=False)
173     file_parser.add_argument(
174         "--db",
175         action="store",
176         required=False,
177         default=str(Path.home() / "workers.db"),
178         help="The database file name"
179     )
180     # Создать основной парсер командной строки.
181     parser = argparse.ArgumentParser("workers")
182     parser.add_argument(
183         "--version",
184         action="version",
185         version="%(prog)s 0.1.0"
186     )
187     subparsers = parser.add_subparsers(dest="command")
188     add = subparsers.add_parser(
189         "add",
190         parents=[file_parser],
191         help="Add a new flight"
192     )
193     add.add_argument(
194         "-d",
195         "--dest",

```

Рисунок 5 – Код программы, продолжение

```

196         action="store",
197         required=True,
198         help="Flight destination"
199     )
200     add.add_argument(
201         "-n",
202         "--flight_num",
203         action="store",
204         help="The flight number"
205     )
206     add.add_argument(
207         "-t",
208         "--type",
209         action="store",
210         type=str,
211         required=True,
212         help="The airplane type"
213     )
214     _ = subparsers.add_parser(
215         "display",
216         parents=[file_parser],
217         help="Display all flights"
218     )
219     select = subparsers.add_parser(
220         "select",
221         parents=[file_parser],
222         help="Select the flights"
223     )
224     select.add_argument(
225         "-T",
226         "--type",
227         action="store",
228         type=str,
229         required=True,
230         help="The required type"
231     )
232     # Выполнить разбор аргументов командной строки.
233     args = parser.parse_args(command_line)
234     # Получить путь к файлу базы данных.

```

select_all()

Рисунок 6 – Код программы, продолжение

```

234 # Получить путь к файлу базы данных.
235 db_path = Path(args.db)
236 create_db(db_path)
237 if args.command == "add":
238     add_flight(db_path, args.dest, args.flight_num, args.type)
239 elif args.command == "display":
240     display_flights(select_all(db_path))
241 elif args.command == "select":
242     display_flights(select_flights(db_path, args.type))
243     pass
244
245
246 if __name__ == "__main__":
247     main()
248

```

Рисунок 7 – Код программы, продолжение

Таблица: flight_numbers

	num_id	num_title
	Фильтр	Фильтр
1	1	HL128
2	2	GN356
3	3	Y4367
4	4	MI194

Рисунок 8 – Изначальный вид тестовой базы данных

Таблица: flights

	flight_id	flight_destination	num_id	airplane_type
	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Hollywood	1	Passenger
2	2	Gtreshd	2	Military
3	3	Yfhds	3	Sanitary

Рисунок 9 – Изначальный вид тестовой базы данных

```

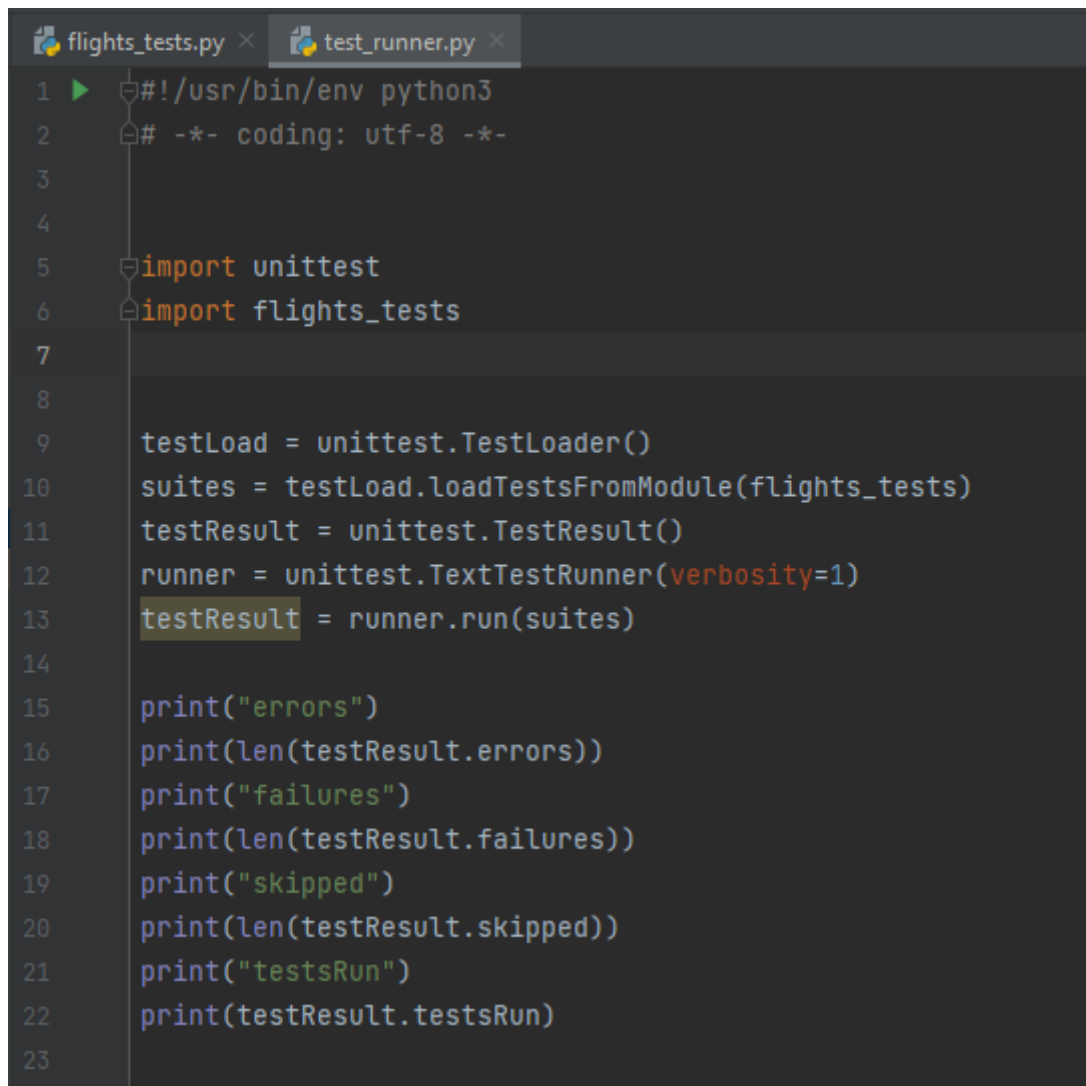
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      import unittest
6      import flights
7
8
9  ▶  class FlightsTests(unittest.TestCase):
10     ▶  def test_select_all_3_rows(self):
11         testlist = [
12             {
13                 'flight_destination': 'Hollywood',
14                 'flight_number': 'HL128',
15                 'airplane_type': 'Passenger'
16             },
17             {
18                 'flight_destination': 'Gtreshd',
19                 'flight_number': 'GH356',
20                 'airplane_type': 'Military'
21             },
22             {
23                 'flight_destination': 'Yfhajs',
24                 'flight_number': 'Y4367',
25                 'airplane_type': 'Sanitary'
26             }
27         ]
28         self.assertEqual(flights.select_all('test_flights.db'), testlist)
29
30     ▶  def test_select_by_type(self):
31         ans = [
32             {
33                 'flight_destination': 'Gtreshd',
34                 'flight_number': 'GH356',
35                 'airplane_type': 'Military'
36             }
37         ]
38         self.assertEqual(
39             flights.select_flights('test_flights.db', 'Military'), ans

```

Рисунок 10 – Содержимое модуля flights_tests.py


```
40     )
41
42     def test_select_all_after_adding(self):
43         dest = "Miami"
44         flight_num = "MI194"
45         type = "Passenger"
46         flights.add_flight('test_flights.db', dest, flight_num, type)
47         testlist = [
48             {
49                 'flight_destination': 'Hollywood',
50                 'flight_number': 'HL128',
51                 'airplane_type': 'Passenger'
52             },
53             {
54                 'flight_destination': 'Gtreshd',
55                 'flight_number': 'GH356',
56                 'airplane_type': 'Military'
57             },
58             {
59                 'flight_destination': 'Yfhhdjs',
60                 'flight_number': 'Y4367',
61                 'airplane_type': 'Sanitary'
62             },
63             {
64                 'flight_destination': 'Miami',
65                 'flight_number': 'MI194',
66                 'airplane_type': 'Passenger'
67             }
68         ]
69         self.assertEqual(flights.select_all('test_flights.db'), testlist)
70
```

Рисунок 11 – Код flights_tests.py, продолжение



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  import unittest
6  import flights_tests
7
8
9  testLoad = unittest.TestLoader()
10 suites = testLoad.loadTestsFromModule(flights_tests)
11 testResult = unittest.TestResult()
12 runner = unittest.TextTestRunner(verbosity=1)
13 testResult = runner.run(suites)
14
15 print("errors")
16 print(len(testResult.errors))
17 print("failures")
18 print(len(testResult.failures))
19 print("skipped")
20 print(len(testResult.skipped))
21 print("testsRun")
22 print(testResult.testsRun)
23
```

Рисунок 12 – Код модуля test_runner.py

```

PS C:\Users\Evil\PycharmProjects\LR#10\ind_task> python test_runner.py
test_flights.db
.test_flights.db
..
-----
Ran 3 tests in 0.078s

OK
errors
0
failures
0
skipped
0
testsRun
3
PS C:\Users\Evil\PycharmProjects\LR#10\ind_task> 

```

Рисунок 13 – Результаты unit-тестирования

Таблица: flights

	flight_id	flight_destination	num_id	airplane_type
	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Hollywood	1	Passenger
2	2	Gtreshd	2	Military
3	3	Yfhjdjs	3	Sanitary
4	4	Miami	4	Passenger

Рисунок 14 – Вид тестовой базы данных после окончания тестирования

Таблица: flight_numbers

	num_id	num_title
	Фильтр	Фильтр
1	1	HL128
2	2	GH356
3	3	Y4367
4	4	MI194

Рисунок 15 – Вид тестовой базы данных после окончания тестирования

3. Ответы на контрольные вопросы

1. Для чего используется автономное тестирование?

Для тестирования функций, классов, методов и т.д. с целью выявления ошибок в работе в этих отдельных единицах общей программы.

2. Какие фреймворки Python получили наибольшее распространение для решения задач автономного тестирования?

В мире Python существуют три framework'а, которые получили наибольшее распространение:

- unittest
- nose
- pytest

unittest

unittest – это framework для тестирования, входящий в стандартную библиотеку языка Python. Его архитектура выполнена в стиле xUnit. xUnit представляет собой семейство framework'ов для тестирования в разных языках программирования, в Java – это JUnit, C# – NUnit и т.д.

nose

Девизом nose является фраза “nose extends unittest to make testing easier”, что можно перевести как “nose расширяет unittest, делая тестирование проще”. nose идеален, когда нужно сделать тесты “по-быстрому”, без предварительного планирования и выстраивания архитектуры приложения с тестами. Функционал nose можно расширять и настраивать с помощью плагинов.

pytest

pytest довольно мощный инструмент для тестирования, и многие разработчики оставляют свой выбор на нем. pytest по “духу” ближе к языку Python нежели unittest. Как было сказано выше, unittest в своей базе – xUnit, что накладывает определенные обязательства при разработке тестов (создание классов-наследников от unittest.TestCase, выполнение определенной процедуры запуска тестов и т.п.). При разработке на pytest ничего этого делать не нужно, вы просто пишете функции, которые должны начинаться с “test_” и используете assert'ы, встроенные в Python (unittest используется свои).

3. Какие существуют основные структурные единицы модуля unittest?

Основными структурными элементами каркаса unittest являются:

Test fixture

Test fixture – обеспечивает подготовку окружения для выполнения тестов, а также организацию мероприятий по их корректному завершению (например, очистка ресурсов). Подготовка окружения может включать в себя создание баз данных, запуск необходим серверов и т.п.

Test case

Test case – это элементарная единица тестирования, в рамках которой проверяется работа компонента тестируемой программы (метод, класс, поведение и т. п.). Для реализации этой сущности используется класс TestCase.

Test suite

Test suite – это коллекция тестов, которая может в себя включать как отдельные test case'ы так и целые коллекции (т.е. можно создавать коллекции коллекций). Коллекции используются с целью объединения тестов для совместного запуска.

Test runner

Test runner – это компонент, который оркестрирует (координирует взаимодействие) запуск тестов и предоставляет пользователю результат их выполнения. Test runner может иметь графический интерфейс, текстовый интерфейс или возвращать какое-то заранее заданное значение, которое будет описывать результат прохождения тестов.

4. Какие существуют способы запуска тестов unittest?

Запуск тестов можно сделать как из командной строки, так и с помощью графического интерфейса пользователя (GUI).

5. Каково назначение класса TestCase?

Он представляет собой класс, который должен являться базовым для всех остальных классов, методы которых будут тестировать те или иные автономные единицы исходной программы. Для того, чтобы метод класса выполнялся как тест, необходимо, чтобы он начинался со слова test. Несмотря на то, что методы framework'a unittest написаны не в соответствии с PEP 8 (ввиду того, что идейно он наследник xUnit), мы все же рекомендуем следовать правилам стиля для Python везде, где это возможно. Поэтому имена тестов будем начинать с префикса test_. Далее, под словом тест будем понимать метод класса-наследника от TestCase, который начинается с префикса test_.

6. Какие методы класса TestCase выполняются при запуске и завершении работы тестов?

К этим методам относятся:

setUp()

Метод вызывается перед запуском теста. Как правило, используется для подготовки окружения для теста.

tearDown()

Метод вызывается после завершения работы теста. Используется для “приборки” за тестом. Заметим, что методы setUp() и tearDown() вызываются для всех тестов в рамках класса, в котором они переопределены. По умолчанию, эти методы ничего не делают. Если их добавить в utest_calc.py, то перед [после] тестов test_add(), test_sub(), test_mul(), test_div() будут выполнены setUp() [tearDown()].

7. Какие методы класса TestCase используются для проверки условий и генерации ошибок?

TestCase класс предоставляет набор assert-методов для проверки и генерации ошибок:

Метод	Описание
<code>assertEqual(a, b)</code>	<code>a == b</code>
<code>assertNotEqual(a, b)</code>	<code>a != b</code>
<code>assertTrue(x)</code>	<code>bool(x) is True</code>
<code>assertFalse(x)</code>	<code>bool(x) is False</code>
<code>assertIs(a, b)</code>	<code>a is b</code>
<code>assertIsNot(a, b)</code>	<code>a is not b</code>
<code>assertIsNone(x)</code>	<code>x is None</code>
<code>assertIsNotNone(x)</code>	<code>x is not None</code>
<code>assertIn(a, b)</code>	<code>a in b</code>
<code>assertNotIn(a, b)</code>	<code>a not in b</code>
<code>assertIsInstance(a, b)</code>	<code>isinstance(a, b)</code>
<code>assertNotIsInstance(a, b)</code>	<code>not isinstance(a, b)</code>

Assert'ы для контроля выбрасываемых исключений и warning'ов:

Метод	Описание
<code>assertRaises(exc, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> вызывает исключение <code>exc</code>
<code>assertRaisesRegex(exc, r, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> вызывает исключение <code>exc</code> , сообщение которого совпадает с регулярным выражением <code>r</code>
<code>assertWarns(warn, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> выдает сообщение <code>warn</code>
<code>assertWarnsRegex(warn, r, fun, *args, **kwargs)</code>	Функция <code>fun(*args, **kwargs)</code> выдает сообщение <code>warn</code> и оно совпадает с регулярным выражением <code>r</code>

Assert'ы для проверки различных ситуаций:

Метод	Описание
<code>assertAlmostEqual(a, b)</code>	<code>round(a-b, 7) == 0</code>
<code>assertNotAlmostEqual(a, b)</code>	<code>round(a-b, 7) != 0</code>
<code>assertGreater(a, b)</code>	<code>a > b</code>
<code>assertGreaterEqual(a, b)</code>	<code>a >= b</code>
<code>assertLess(a, b)</code>	<code>a < b</code>
<code>assertLessEqual(a, b)</code>	<code>a <= b</code>
<code>assertRegex(s, r)</code>	<code>r.search(s)</code>
<code>assertNotRegex(s, r)</code>	<code>not r.search(s)</code>
<code>assertCountEqual(a, b)</code>	<code>a</code> и <code>b</code> имеют одинаковые элементы (порядок неважен)

Типо-зависимые `assert`'ы, которые используются при вызове `assertEqual()`. Приводятся на тот случай, если необходимо использовать конкретный метод.

Метод	Описание
<code>assertMultiLineEqual(a, b)</code>	строки (strings)
<code>assertSequenceEqual(a, b)</code>	последовательности (sequences)
<code>assertListEqual(a, b)</code>	списки (lists)
<code>assertTupleEqual(a, b)</code>	кортежи (tuples)
<code>assertSetEqual(a, b)</code>	множества или неизменяемые множества (frozensets)
<code>assertDictEqual(a, b)</code>	словари (dicts)

Дополнительно хотелось бы отметить метод `fail()`.

`fail(msg=None)`

Этот метод сигнализирует о том, что произошла ошибка в тесте.

8. Какие методы класса `TestCase` позволяют собирать информацию о самом тесте?

`countTestCases()`

Возвращает количество тестов в объекте класса-наследника от `TestCase`.

`id()`

Возвращает строковый идентификатор теста. Как правило это полное имя метода, включающее имя модуля и имя класса.

`shortDescription()`

Возвращает описание теста, которое представляет собой первую строку `docstring`'а метода, если его нет, то возвращает `None`.

9. Каково назначение класса `TestSuite`? Как осуществляется загрузка тестов?

Класс `TestSuite` используется для объединения тестов в группы, которые могут включать в себя как отдельные тесты, так и заранее созданные группы. Помимо этого, `TestSuite` предоставляет интерфейс, позволяющий `TestRunner`'у, запускать тесты.

Метод `*run(result)*` запускает тесты из данной группы.

Начнем с класса `TestLoader`. Этот класс используется для создания групп из классов и модулей. Среди методов `TestLoader` можно выделить:

`loadTestsFromTestCase(testCaseClass)`, возвращающий группу со всеми тестами из класса `testCaseClass`. Напоминаем, что под тестом понимается модуль, начинающийся со слова “test”. Используя этот метод, можно создать список групп тестов, где каждая группа создается на базе классов-наследников от `TestCase`, объединенных предварительно в список.

10. Каково назначение класса `TestResult`?

Класс `TestResult` используется для сбора информации о результатах прохождения тестов.

11. Для чего может понадобиться пропуск отдельных тестов?

Во избежание ошибок тестирования, так как некоторые тесты могут давать заведомо неправильный результат в зависимости от какого-либо условия. Для этого такие тесты необходимо пропускать.

12. Как выполняется безусловный и условных пропуск тестов? Как выполнить пропуск класса тестов?

Безусловный пропуск: `@unittest.skip(reason)` записывается перед объявлением теста.

Условный пропуск:

- 1) `@unittest.skipIf(condition, reason)` – Тест будет пропущен, если условие (`condition`) истинно.
- 2) `@unittest.skipUnless(condition, reason)` – Тест будет пропущен если, условие (`condition`) не истинно.

Пропуск класса тестов: `@unittest.skip(reason)` записывается перед объявлением класса.

13. Самостоятельно изучить средства по поддержке тестов `unittest` в `PyCharm`. Приведите обобщенный алгоритм проведения тестирования с помощью `PyCharm`.

В `PyCharm` есть встроенная поддержка `unit` тестов, которая позволяет создавать шаблон класса для тестирования и его дальнейшей настройки.

- 1) Необходимо создать класс для тестирования.
- 2) Написание кода тестов в классе для тестирования частей программы.
- 3) Запуск тестов
- 4) `Debug` тестов при необходимости.

5) Автоматизация тестов. `PyCharm` поддерживает автоматизацию тестов – установив её, вы можете сфокусироваться в написании кода самой

программы, а IDE будет в автоматическом режиме проводить тестирование по мере изменения кода.