Азиев Андрей
ИУ5-36б

Рубежный контроль №2

Текст программы

tasks.py

```
def first_task(conductors, orchestras):
    r = [(
    conductor.fio, orchestra.name)
    for conductor in conductors
        for orchestra in orchestras
            if (conductor.orchestra_id == orchestra.id)
                and (conductor.fio[-2:] == "ни")]
    return r
def second_task(conductors, orchestras):
    r = [[_.name, 0, 0] for _ in orchestras]
    for conductor in conductors:
        r[conductor.orchestra_id - 1][1] += conductor.years_of_experience
        r[conductor.orchestra_id - 1][2] += 1
    r = [(_[0], _[1]/_[2]) for _ in r]
    r.sort(key=lambda r: r[1])
    return r

def third_task(conductors, orchestras, many_to_many):
    r = [(_.name, []) for _ in orchestras]
    connections = [(orchestras[connection.orchestra_id-1].name, conductors[connection.conductor_id-1].fio) for connection in
many_to_many]
    for _ in connections:
        for elm in r:
            if elm[0] == _[0]:
                elm[1].append(_[1])
    to_be_del = []
    for i in range (len(r)):
        if r[i][0][0] != "R":
            to_be_del.append(i)
    to_be_del.reverse()
    for i in to_be_del:
        del r[i]
    return r

if __name__ == '__main__':
```

```
main()
```

tests.py

```
import unittest

from src.conductor import *
from src.orchestra import *
from src.connection import *

import tasks

class TestRK(unittest.TestCase):

    orchestras = [
        Orchestra(1, "Royal Concertgebouw Orchestra", "Amsterdam"),
        Orchestra(2, "Berlin Philharmonic Orchestra", "Berlin"),
        Orchestra(3, "Los Angeles Philharmonic", "Los-Angeles")
    ]
    conductors = [
        Conductor(1, "Томас Бичем", 1, 10),
        Conductor(2, "Карло Мария Джулини", 2, 15),
        Conductor(3, "Артуро Тосканини", 3, 16),
        Conductor(4, "Вильгельм Фуртвенглер", 2, 19),
        Conductor(5, "Николаус Арнонкур", 1, 12)
    ]
    many_to_many = [
        ConductorOrchestra(3, 1),
        ConductorOrchestra(2, 1),
        ConductorOrchestra(1, 2),
        ConductorOrchestra(1, 3),
        ConductorOrchestra(4, 2)
    ]
    def test_1(self):
        self.assertEqual(tasks.first_task(self.conductors, self.orchestras), [('Карло Мария Джулини', 'Berlin Philharmonic Orchestra'), ('Артуро Тосканини', 'Los Angeles Philharmonic')], 'wrong')
    def test_2(self):
        self.assertEqual(tasks.second_task(self.conductors, self.orchestras), [('Royal Concertgebouw Orchestra', 11.0), ('Los Angeles Philharmonic', 16.0), ('Berlin Philharmonic Orchestra', 17.0)], 'wrong')
    def test_3(self):
        self.assertEqual(tasks.third_task(self.conductors, self.orchestras, self.many_to_many), [('Royal Concertgebouw Orchestra', ['Артуро Тосканини', 'Карло Мария Джулини'])], 'wrong')

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

Результат программы

```
test_1 (tests.TestRK.test_1) ... ok
test_2 (tests.TestRK.test_2) ... ok
test_3 (tests.TestRK.test_3) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```