Код программы

main.py

```
class File:
    def __init__(self, id : int, name : str, extension : str, size : int,
catalog_id : int):
        self.id = id
        self.name = name
        self.extension = extension
        self.size = size
        self.catalog_id = catalog_id

class Catalog:
    def __init__(self, id : int, name : str, path : str):
        self.id = id
        self.name = name
        self.path = path

class FileCatalog:
    def __init__(self, id : int, catalog_id : int, file_id : int):
        self.id = id
        self.file_id = file_id
        self.catalog_id = catalog_id

catalogs = [
    Catalog(1, 'материалы по предмету правоведение', 'D:\\edu\\jurisprudence\\'),
    Catalog(2, 'материалы по предмету твимс', 'D:\\edu\\tvims\\'),
    Catalog(3, 'фото', 'D:\\photos\\'),
    Catalog(4, 'материалы по предмету пикяп', 'D:\\edu\\pikyap\\'),
    Catalog(5, 'загрузки', 'C:\\Users\\Roma\\Downloads\\'),
]

files = [
    File(1, 'домашнее задание 1', 'docx', 266000, 1),
    File(2, 'домашнее задание 2', 'docx', 150000, 1),
    File(3, 'домашнее задание 1', 'png', 100000, 2),
    File(4, 'домашнее задание 2', 'jpg', 266000, 2),
    File(5, 'лекция 1', 'pdf', 520000, 2),
    File(6, 'image1', 'jpg', 31000, 3),
    File(7, 'image2', 'jpg', 46000, 3),
    File(8, 'лаб 1', 'py', 12000, 4),
    File(9, 'лаб 1', 'pdf', 67000, 4),
    File(10, 'лаб 2', 'cpp', 15000, 4),
    File(11, 'лаб 2', 'pdf', 76000, 4),
    File(12, 'deusex2001', 'torrent', 1500, 5),
    File(13, 'даталогическая модель', 'drawio', 335000, 5),
    File(14, 'ghostrunner2', 'exe', 546000789, 5),
]
```

```
FilesCatalogs = [
    FileCatalog(1, 1, 1),
    FileCatalog(2, 1, 2),
    FileCatalog(3, 1, 5),
    FileCatalog(4, 2, 3),
    FileCatalog(5, 2, 4),
    FileCatalog(6, 2, 5),
    FileCatalog(7, 2, 13),
    FileCatalog(8, 3, 6),
    FileCatalog(9, 3, 7),
    FileCatalog(10, 3, 3),
    FileCatalog(11, 3, 4),
    FileCatalog(12, 4, 8),
    FileCatalog(13, 4, 9),
    FileCatalog(14, 4, 10),
    FileCatalog(15, 4, 11),
    FileCatalog(16, 4, 7),
    FileCatalog(17, 5, 12),
    FileCatalog(18, 5, 13),
    FileCatalog(19, 5, 14),
    FileCatalog(20, 5, 1),
    FileCatalog(21, 5, 2),
]

e1_filter = 'предмет' # строка для фильтрации в задании 1
e3_filter = 'л' # символ для фильтрации в задании 3

def task1():
    print('Задание E1:')
    answer = ''
    for _catalog in catalogs:
        if e1_filter in _catalog.name:
            answer += f'Каталог: {_catalog.name} Путь: {_catalog.path}\n'
            filtered_files = list(filter(lambda x: (x.catalog_id == _catalog.id),
files))
            answer += '    ' + '\n    '.join([f'Имя файла: {i.name}.{i.extension}
Размер: {i.size}' for i in filtered_files])
            answer += '\n'
    return answer[:-1]

def task2():
    print('Задание E2:')
    files_count_in_catalog = lambda _catalog: len([_file for _file in files if
_file.catalog_id == _catalog.id])
    size_average_in_catalog = lambda _catalog: round(sum([_file.size for _file in
files if _file.catalog_id == _catalog.id]) / files_count_in_catalog(_catalog), 2)
    catalogs_average_size = {
        _catalog: size_average_in_catalog(_catalog)
        for _catalog in catalogs
    }
```

```
    sorted_catalog_average_size = dict(sorted(catalogs_average_size.items(),
key=lambda item: item[1]))
    answer = '\n'.join([f'Каталог: {_catalog.name} {_catalog.path} Средний
размер: {average_size}' for _catalog, average_size in
sorted_catalog_average_size.items()])
    return answer

def task3():
    print('Задание E3:')
    answer = ''
    filtered_files = list(filter(lambda x: (x.name[0] == e3_filter), files))
    for _file in filtered_files:
        file_catalogs_ids = [filecatalogs.catalog_id for filecatalogs in
FilesCatalogs if _file.id == filecatalogs.file_id]
        answer += f'Файл: {_file.name}.{_file.extension} Размер:
{_file.size}\nКаталоги:\n    ' + '\n    '.join([f'{_catalog.name}
{_catalog.path}' for _catalog in catalogs if _catalog.id in file_catalogs_ids])
    return answer

def main():
    print(task1())
    print(task2())
    print(task3())

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

tests.py

```
import unittest

from main import *

class TestSolutions(unittest.TestCase):
    def setUp(self):
        self.catalogs = [
            Catalog(1, 'материалы по предмету правоведение',
'D:\\edu\\jurisprudence\\'),
            Catalog(2, 'материалы по предмету твимс', 'D:\\edu\\tvims\\'),
            Catalog(3, 'фото', 'D:\\photos\\'),
            Catalog(4, 'материалы по предмету пикяп', 'D:\\edu\\pikyap\\'),
            Catalog(5, 'загрузки', 'C:\\Users\\Roma\\Downloads\\'),
        ]
        self.files = [
            File(1, 'домашнее задание 1', 'docx', 266000, 1),
            File(2, 'домашнее задание 2', 'docx', 150000, 1),
            File(3, 'домашнее задание 1', 'png', 100000, 2),
            File(4, 'домашнее задание 2', 'jpg', 266000, 2),
            File(5, 'лекция 1', 'pdf', 520000, 2),
            File(6, 'image1', 'jpg', 31000, 3),
            File(7, 'image2', 'jpg', 46000, 3),
            File(8, 'лаб 1', 'py', 12000, 4),
```

```
            File(9, 'лаб 1', 'pdf', 67000, 4),
            File(10, 'лаб 2', 'cpp', 15000, 4),
            File(11, 'лаб 2', 'pdf', 76000, 4),
            File(12, 'deusex2001', 'torrent', 1500, 5),
            File(13, 'даталогическая модель', 'drawio', 335000, 5),
            File(14, 'ghostrunner2', 'exe', 546000789, 5),
        ]
        self.FilesCatalogs = [
            FileCatalog(1, 1, 1),
            FileCatalog(2, 1, 2),
            FileCatalog(3, 1, 5),
            FileCatalog(4, 2, 3),
            FileCatalog(5, 2, 4),
            FileCatalog(6, 2, 5),
            FileCatalog(7, 2, 13),
            FileCatalog(8, 3, 6),
            FileCatalog(9, 3, 7),
            FileCatalog(10, 3, 3),
            FileCatalog(11, 3, 4),
            FileCatalog(12, 4, 8),
            FileCatalog(13, 4, 9),
            FileCatalog(14, 4, 10),
            FileCatalog(15, 4, 11),
            FileCatalog(16, 4, 7),
            FileCatalog(17, 5, 12),
            FileCatalog(18, 5, 13),
            FileCatalog(19, 5, 14),
            FileCatalog(20, 5, 1),
            FileCatalog(21, 5, 2),
        ]
        self.test_word = 'предмет'
        self.test_letter = 'л'
    def test_task1_solution(self):
        result = task1()
        self.assertEqual(
            result,
            """Каталог: материалы по предмету правоведение Путь:
D:\\edu\\jurisprudence\\
    Имя файла: домашнее задание 1.docx Размер: 266000
    Имя файла: домашнее задание 2.docx Размер: 150000
Каталог: материалы по предмету твимс Путь: D:\\edu\\tvims\\
    Имя файла: домашнее задание 1.png Размер: 100000
    Имя файла: домашнее задание 2.jpg Размер: 266000
    Имя файла: лекция 1.pdf Размер: 520000
Каталог: материалы по предмету пикяп Путь: D:\\edu\\pikyap\\
    Имя файла: лаб 1.py Размер: 12000
    Имя файла: лаб 1.pdf Размер: 67000
    Имя файла: лаб 2.cpp Размер: 15000
    Имя файла: лаб 2.pdf Размер: 76000"""
        )
```

```
    def test_task2_solution(self):
        result = task2()
        self.assertEqual(
            result,
            """Каталог: фото D:\\photos\\ Средний размер: 38500.0
Каталог: материалы по предмету пикяп D:\\edu\\pikyap\\ Средний размер: 42500.0
Каталог: материалы по предмету правоведение D:\\edu\\jurisprudence\\ Средний
размер: 208000.0
Каталог: материалы по предмету твимс D:\\edu\\tvims\\ Средний размер: 295333.33
Каталог: загрузки C:\\Users\\Roma\\Downloads\\ Средний размер: 182112429.67"""
        )

    def test_task3_solution(self):
        result = task3()
        self.assertEqual(
            result,
            """Файл: лекция 1.pdf Размер: 520000
Каталоги:
    материалы по предмету правоведение D:\\edu\\jurisprudence\\
    материалы по предмету твимс D:\\edu\\tvims\\Файл: лаб 1.py Размер: 12000
Каталоги:
    материалы по предмету пикяп D:\\edu\\pikyap\\Файл: лаб 1.pdf Размер: 67000
Каталоги:
    материалы по предмету пикяп D:\\edu\\pikyap\\Файл: лаб 2.cpp Размер: 15000
Каталоги:
    материалы по предмету пикяп D:\\edu\\pikyap\\Файл: лаб 2.pdf Размер: 76000
Каталоги:
    материалы по предмету пикяп D:\\edu\\pikyap\\"""
        )

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

Результаты работы

```
Ran 3 tests in 0.003s

OK

Ran 3 tests in 0.006s

FAILED (failures=1)
```