

# Лабораторная работа №4

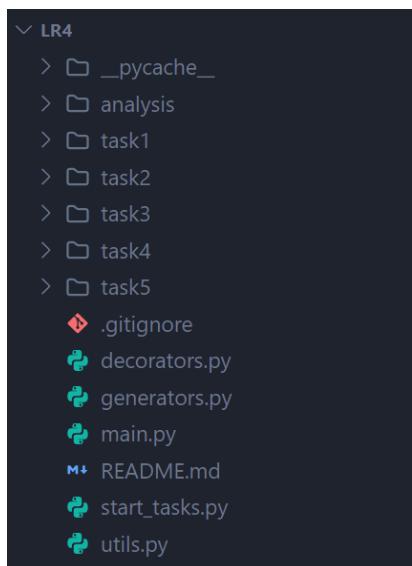
**Тема:** работа с файлами, классами, сериализаторами, регулярными выражениями и стандартными библиотеками.

**Цель:** освоить базовый синтаксис языка Python, приобрести навыки работы с файлами, классами, сериализаторами, регулярными выражениями и стандартными библиотеками и закрепить их на примере разработки интерактивных приложений.

**Выполнил:** студент группы 253502 Красёв П.А.

Вариант 11.

Структура лабораторной работы:



## Файл main.py:

```
1 from start_tasks import TasksStarter
2
3 # Lab4. Working with files, classes, serializers, regular expressions and standard Libraries.
4 # Performed by a student of group 253502, Krasyov Pavel.
5 # Fulfillment date: 25.03.2024
6
7
8 def main():
9     starter = TasksStarter()
10    while True:
11        try:
12            task = int(input(
13                'Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: '))
14            if task < 0 or task > 6:
15                raise Exception
16        except:
17            print(
18                'Incorrect input, try again! Input only integer numbers between 0 and 6!\n')
19            continue
20        if task == 1:
21            starter.runTask1()
22            continue
23        elif task == 2:
24            starter.runTask2()
25            continue
26        elif task == 3:
27            starter.runTask3()
28            continue
29        elif task == 4:
30            starter.runTask4()
31            continue
32        elif task == 5:
33            starter.runTask5()
34            continue
35        elif task == 6:
36            starter.runAnalisyis()
37            continue
38        elif task == 0:
39            break
40    print('Program finished!')
41
42
43 if __name__ == '__main__':
44     main()
```

## Функция-генератор матрицы для задания 5:

```
1 import numpy as np
2
3
4 def matrixGenerator(rows: int, columns: int):
5     return np.random.randint(-150, 150, size=(rows, columns))
6
```

Декоратор для функций запуска заданий:

```
1 from utils import stopFunction
2
3
4 def RunTask(Class):
5     def RunningDecorator(func):
6         def RunLoop(self):
7             print(func.__doc__)
8             obj = Class()
9             while True:
10                 res = func(self, obj)
11                 if res is not None or stopFunction('finish task') != "0":
12                     break
13                 print('Task exit')
14             return RunLoop
15     return RunningDecorator
16
```

Функции проверки ввода:

```
1 from matplotlib import pyplot as plt
2
3
4 def stopFunction(action: str = 'continue') -> str:
5     ''' Function stops running task. '''
6     return input(f'Do you want to {action}? (0 -> No | anything else -> Yes): ')
7
8
9 def validateNumberInput(inputMessage: str, errorMessage: str, numberType, condition=lambda x: type(x) == str) -> int:
10     ''' Function validates input of a number with a given type and condition. '''
11     while True:
12         try:
13             N = numberType(
14                 input(f'Enter {inputMessage}: '))
15             if condition(N):
16                 print(f'Wrong input -> ({errorMessage})! Try again.\n')
17                 continue
18             except ValueError:
19                 print('Wrong input! Try again.\n')
20                 continue
21         break
22     return N
23
24
25 def validateColorInput(figure: str) -> str:
26     ''' Function validates input of figure color. '''
27     while True:
28         color = input(f'Input {figure} color: ')
29         try:
30             plt.plot([], [], color=color)
31         except ValueError:
32             print('Color is not available!\n')
33             continue
34         break
35     return color
```

**Задание 1.** Исходные данные представляют собой словарь. Необходимо поместить их в файл, используя сериализатор. Организовать считывание данных, поиск, сортировку в соответствии с индивидуальным заданием. Обязательно использовать классы.

Реализуйте два варианта: 1)формат файлов CSV; 2)модуль pickle;

Вам дан словарь, состоящий из слов, расположенных парами. Каждое слово является синонимом к парному ему слову. Все слова в словаре различны. Для последнего слова из словаря определите его синоним. Выведите информацию о слове, введенном с клавиатуры.

Функция запускающая задание:

```
@RunTask(Task1)
def runTask1(self, task: Task1) -> str | None:
    ''' Task1\nA dictionary consisting of words arranged in pairs is given. Each word is a synonym for its paired word. All words in the dictionary are different. For the last word in the dictionary, determine its synonym.
    Display information about the word entered from the keyboard. '''
    var = validateNumberInput('type (1 -> csv file | 2 -> pickle module | 0 -> Exit function)',
                             '1 -> csv file | 2 -> pickle module | 0 -> Exit function', int,
                             condition=lambda x: x != 0 and x != 1 and x != 2)
    if var == 1:
        dictionary = CsvDictionary(f'{task.path}csv')
    elif var == 2:
        dictionary = PickleDictionary(f'{task.path}pkl')
    else:
        return 'stop'
    task(dictionary)
```

Исходный словарь и родительский класс словарей:

```
1  synonyms_dict = dict([["быстрый", "скорый"],
2                         ["медленный", "несспешный"],
3                         ["умный", "интеллектуальный"],
4                         ["глупый", "неразумный"],
5                         ["смелый", "отважный"],
6                         ["трусливый", "боязливый"]])
7
8
9  class BaseDictionary():
10     def __init__(self, FILENAME: str) -> None:
11         ''' Inits class object. '''
12         self.filename = FILENAME
13
14     def getDictionary(self) -> dict[str, str]:
15         ''' Method returns dictionary from the file. '''
16         pass
17
18     def getLastSynonym(self) -> str:
19         ''' Method returns last word-synonym from the dictionary. '''
20         return list(self.getDictionary().values())[-1]
21
22     def getWordInfo(self, word: str) -> None:
23         ''' Method returns info about the word according on the file. '''
24         dictionary = self.getDictionary()
25         values: list[str] = list(dictionary.values())
26         if word in values:
27             try:
28                 print(
29                     f'Pair: {list(dictionary.keys())[values.index(word)]} - {word}')
30             except ValueError:
31                 print('Word isn\'t found!')
32             finally:
33                 return
34         synonym: str | None = dictionary.get(word)
35         print(
36             f'Pair: {word} - {synonym}' if synonym is not None else 'Word isn\'t found!')
```

Класс csv словаря:

```
1 import csv
2 from task1.baseDictionary import BaseDictionary, synonyms_dict
3
4
5 class CsvDictionary(BaseDictionary):
6     def __init__(self, FILENAME: str) -> None:
7         ''' Inits class object. '''
8         super().__init__(FILENAME)
9         with open(FILENAME, "w", newline='') as file:
10             csv.writer(file).writerows(
11                 {k: synonyms_dict[k] for k in sorted(synonyms_dict)}.items())
12
13     def getDictionary(self) -> dict[str, str]:
14         ''' Method returns dictionary from csv file. '''
15         with open(self.filename, 'r', newline='\n') as file:
16             reader = csv.reader(file)
17             synonyms: dict[str, str] = dict()
18             for row in reader:
19                 synonyms[row[0]] = row[1]
20             return synonyms
21
```

Класс pickle словаря:

```
1 import pickle
2 from task1.baseDictionary import BaseDictionary, synonyms_dict
3
4
5 class PickleDictionary(BaseDictionary):
6     def __init__(self, FILENAME: str) -> None:
7         ''' Inits class object. '''
8         super().__init__(FILENAME)
9         with open(FILENAME, 'wb') as file:
10             pickle.dump({k: synonyms_dict[k]
11                         for k in sorted(synonyms_dict)}, file)
12
13     def getDictionary(self) -> dict[str, str]:
14         ''' Method returns dictionary from pkl file. '''
15         with open(self.filename, 'rb') as file:
16             return pickle.load(file)
17
```

### Общий класс задания 1:

```
1 from task1.csvDictionary import CsvDictionary
2 from task1.pickleDictionary import PickleDictionary
3 from utils import stopFunction
4
5
6 class Task1():
7     def __init__(self) -> None:
8         ''' Inits object of the class. '''
9         self.path = './task1/file.'
10
11     def __call__(self, dictionary: CsvDictionary | PickleDictionary) -> None:
12         ''' Runs class methods. '''
13         self.printDictionary(dictionary)
14         self.getLastWord(dictionary)
15         self.getWordInfo(dictionary)
16
17     def printDictionary(self, dictionary: CsvDictionary | PickleDictionary) -> None:
18         ''' Method returns loaded from file dictionary. '''
19         print('Dictionary:')
20         for words in dictionary.getDictionary().items():
21             print(f'{words[0]} - {words[1]}' )
22         print()
23
24     def getLastWord(self, dictionary: CsvDictionary | PickleDictionary) -> None:
25         ''' Method returns last word-synonym from dictionary. '''
26         print('Last word-synonym: ')
27         print(f'{dictionary.getLastSynonym()}')
28         print()
29
30     def getWordInfo(self, dictionary: CsvDictionary | PickleDictionary) -> None:
31         ''' Method returns word's synonym from the dictionary if it's possible. '''
32         while True:
33             word = input('Enter word to find: ')
34             dictionary.getWordInfo(word)
35             print()
36             if stopFunction() == '0':
37                 break
```

## Пример использования:

```
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 1
Task1
A dictionary consisting of words arranged in pairs is given. Each word is a synonym for its paired word. All words in the dictionary are different. For the last word in the dictionary, determine its synonym. Display information about the word entered from the keyboard.
Enter type (1 -> csv file | 2 -> pickle module | 0 -> Exit function): 1
Dictionary:
быстрый - скорый
глупый - неразумный
медленный - неспешный
смелый - отважный
трусливый - боязливый
умный - интеллектуальный

Last word-synonym:
интеллектуальный

Enter word to find: глупый
Pair: глупый - неразумный

Do you want to continue? (0 -> No | anything else -> Yes): 0
Do you want to finish task? (0 -> No | anything else -> Yes): 0
Enter type (1 -> csv file | 2 -> pickle module | 0 -> Exit function): 2
Dictionary:
быстрый - скорый
глупый - неразумный
медленный - неспешный
смелый - отважный
трусливый - боязливый
умный - интеллектуальный

Last word-synonym:
интеллектуальный

Enter word to find: боязливый
Pair: трусливый - боязливый

Do you want to continue? (0 -> No | anything else -> Yes): 0
Do you want to finish task? (0 -> No | anything else -> Yes): yes
Task exit
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 0
Program finished!
PS D:\IGI\253502_KRASOV_11\IGI\LR4>
```

**Задание 2.** В соответствии с заданием своего варианта составить программу для анализа текста. Считать из исходного файла текст. Используя регулярные выражения получить искомую информацию (см. условие), вывести ее на экран и сохранить в другой файл. Заархивировать файл с результатом с помощью модуля zipfile и обеспечить получение информации о файле в архиве. Также выполнить общее задание – определить и сохранить в файл с результатами:

- количество предложений в тексте;
- количество предложений в тексте каждого вида отдельно (повествовательные, вопросительные и побудительные);
- среднюю длину предложения в символах (считываются только слова);
- среднюю длину слова в тексте в символах;
- количество смайликов в заданном тексте. Смайликом будем считать последовательность символов, удовлетворяющую условиям:
  - первым символом является либо «;» (точка с запятой) либо «::» (двоеточие) ровно один раз;
  - далее может идти символ «-» (минус) сколько угодно раз (в том числе символ минус может идти ноль раз);
  - в конце обязательно идет некоторое количество (не менее одной) одинаковых скобок из следующего набора: «(», «)», «[», «]»;
  - внутри смайлика не может встречаться никаких других символов.

Найти в тексте:

- вывести все слова, включающие символы, лежащих в диапазоне от 'a' до 'o' и цифры;

- проверить, является ли заданная строка шестизначным числом, записанным в десятичной системе счисления без нулей в старших разрядах. Примеры правильных выражений: 123456, 234567. Примеры неправильных выражений: 1234567, 12345;
- определить количество слов, заключенных в кавычки;
- определить, сколько раз повторяется каждая буква;
- вывести в алфавитном порядке все словосочетания, отделенные запятыми.

Функция, запускающая задание:

```
@RunTask(Task2)
def runTask2(self, task: Task2) -> None:
    ''' Task2\nRead text from the source file. Using regular expressions, obtain the required information display it on the screen and save it to another file. Archive the result file using the zipfile module and ensure that information about the file is retrieved in the archive. '''
    task()
```

Абстрактный класс анализатора текста:

```
1  from abc import ABC
2
3
4  class Analiser(ABC):
5      def countSentences(self) -> int:
6          ''' Method returns amount of sentences. '''
7          pass
8
9      def countTypesSentences(self) -> tuple[int, int, int]:
10         ''' Method returns amount of each type sentence. '''
11         pass
12
13     def avgSentenceLen(self) -> float:
14         ''' Method returns average sentence length. '''
15         pass
16
17     def avgWordLen(self) -> float:
18         ''' Method return average word length. '''
19         pass
20
21     def countSmiles(self) -> int:
22         ''' Method returns ammount of smiles in the text. '''
23         pass
24
25     def getMatchWords(self) -> list[str]:
26         ''' Method returns words with letters from \'a\' to \'o\' or digits. '''
27         pass
28
29     def checkStr(self, input_string: str) -> bool:
30         ''' Method checks if input string is 6-digits number in decimal system without leading zeros. '''
31         pass
32
33     def countQuotesWords(self) -> int:
34         ''' Method returns amount of words in quotes. '''
35         pass
36
37     def countLetters(self) -> dict[str, int]:
38         ''' Method returns amount of each letter in text. '''
39         pass
40
41     def getPhrases(self) -> list[str]:
42         ''' Method returns phrases in commas from the text. '''
43         pass
```

Класс анализатора текста, реализующий методы абстрактного класса:

```
1 import re
2 from collections import Counter
3 from task2.baseAnaliser import Analiser
4
5
6 class TextAnaliser(Analiser):
7     def __init__(self, text: str):
8         ''' Inits class object. '''
9         self.text = text
10        self.sentences = re.split(r'(?<=[.!?])\s', text)
11        self.words = re.findall(r'\w+', self.text)
12
13    def countSentences(self) -> int:
14        ''' Method returns amount of sentences. '''
15        return len(self.sentences)
16
17    def countTypesSentences(self) -> tuple[int, int, int]:
18        ''' Method returns amount of each type sentence. '''
19        return (len([1 for sent in self.sentences if sent.endswith('.')]),
20               len([1 for sent in self.sentences if sent.endswith('?')]),
21               len([1 for sent in self.sentences if sent.endswith('!')]))
22
23    def avgSentenceLen(self) -> float:
24        ''' Method returns average sentence length. '''
25        sentence_lengths = [len(word) for word in self.words]
26        return round(sum(sentence_lengths) / len(self.sentences), 3) if self.sentences else 0
27
28    def avgWordLen(self) -> float:
29        ''' Method return average word length. '''
30        words_lengths = [len(word) for word in self.words]
31        return round(sum(words_lengths) / len(self.words), 3) if self.words else 0
32
33    def countSmiles(self) -> int:
34        ''' Method returns ammount of smiles in the text. '''
35        return len(re.findall(r'[:;]-*[()\\[]]+', self.text))
36
37    def getMatchWords(self) -> list[str]:
38        ''' Method returns words with letters from \'a\' to \'o\' or digits. '''
39        return re.findall(r'\b\w*([a-o]+\w*\d+|\d+\w*[a-o]+\w*\b', self.text)
40
41    def checkStr(self, input_string: str) -> bool:
42        ''' Method checks if input string is 6-digits number in decimal system without leading zeros. '''
43        return bool(re.match(r'^[1-9]\d{5}$', input_string))
44
45    def countQuotesWords(self) -> int:
46        ''' Method returns amount of words in quotes. '''
47        return len(re.findall(r'\s(\"|'|«)(.+?)(\"|'|«)', self.text))
48
49    def countLetters(self) -> dict[str, int]:
50        ''' Method returns amount of each letter in text. '''
51        return Counter(re.findall(r'[a-z]', self.text.lower()))
52
53    def getPhrases(self) -> list[str]:
54        ''' Method returns phrases in commas from the text. '''
55        return sorted([ph[0][2:-1].lower() for ph in re.findall(r'(\.,(\s+\w+)\{2,\},)', self.text)])
```

## Общий класс задания 2:

```
1 from task2.textAnaliser import TextAnaliser
2 from task2.baseAnaliser import Analiser
3 from utils import validateNumberInput
4 from zipfile import ZIP_DEFLATED, ZipFile
5
6
7 class Task2(Analiser):
8     _border: str = ('*' * 80) + "\n"
9
10    def __init__(self) -> None:
11        ''' Inits class object. '''
12        self.path = './task2/'
13        self.results: list[str] = list()
14        with open(f'{self.path}file.txt', 'r', encoding='utf-8') as file:
15            text = file.read()
16        self.analiser = TextAnaliser(text)
17
18    def __call__(self) -> None:
19        ''' Runs class methods. '''
20        self.results.clear()
21        self.countSentences()
22        self.countTypesSentences()
23        self.avgSentenceLen()
24        self.avgWordLen()
25        self.countSmiles()
26        self.getMatchWords()
27        self.checkStr()
28        self.countQuotesWords()
29        self.countLetters()
30        self.countPhrases()
31        self.saveResults()
32        self.zipFile()
33
34    def countSentences(self) -> None:
35        ''' Method returns amount of sentences in the text. '''
36        self.results.append(
37            f'Sentenses amount: {self.analiser.countSentences()}\n' + self._border)
38
39    def countTypesSentences(self) -> None:
40        ''' Method returns amount of each type sentence. '''
41        amount = self.analiser.countTypesSentences()
42        self.results.append(f'''Declarative sentenses amount: {amount[0]}  

43 Interrogative sentences amount: {amount[1]}  

44 Incentive sentenses amount: {amount[2]}\n''' + self._border)
45
46    def avgSentenceLen(self) -> None:
47        ''' Method returns average sentence length. '''
48        self.results.append(
49            f'Average sentence length: {self.analiser.avgSentenceLen()}\n' + self._border)
```

```

51     def avgWordLen(self) -> None:
52         ''' Method return average word length. '''
53         self.results.append(
54             f'Average word length: {self.analiser.avgWordLen()}\n' + self._border)
55
56     def countSmiles(self) -> None:
57         ''' Method returns amount of smiles in the text. '''
58         self.results.append(
59             f'Amount of smiles: {self.analiser.countSmiles()}\n' + self._border)
60
61     def getMatchWords(self) -> None:
62         ''' Method returns words with letters from \'a\' to \'o\' or digits. '''
63         result_str = 'Words with letters from \'a\' to \'o\' and digits:\n'
64         for word in self.analiser.getMatchWords():
65             result_str += word + "\n"
66         self.results.append(result_str + self._border)
67
68     def checkStr(self) -> None:
69         ''' Method checks if input string is 6-digits number in decimal system without leading zeros. '''
70         input_string = str(validateNumberInput(
71             'integer number string', '', int))
72         res = '' if self.analiser.checkStr(input_string) else ' not'
73         self.results.append(
74             f'String {input_string} is{res} valid\n' + self._border)
75         print()
76
77     def countQuotesWords(self) -> None:
78         ''' Method returns amount of words in quotes. '''
79         self.results.append(
80             f'Amount of words in quotes: {self.analiser.countQuotesWords()}\n' + self._border)
81
82     def countLetters(self) -> None:
83         ''' Method returns amount of each letter in text. '''
84         result_str = 'Amount of each letter:\n'
85         for res in self.analiser.countLetters().items():
86             result_str += f'{res[0]} - {res[1]}\n'
87         self.results.append(result_str + self._border)
88
89     def countPhrases(self) -> None:
90         ''' Method returns phrases in commas from the text. '''
91         result_str = 'Phrases in commas:\n'
92         for phrase in self.analiser.getPhrases():
93             result_str += phrase + "\n"
94         self.results.append(result_str + self._border)
95
96     def saveResults(self) -> None:
97         ''' Method saves results to a file. '''
98         print('Analised text:')
99         print(self.analiser.text)
100        print("\nResults:")
101        with open(f'{self.path}results.txt', 'w', newline='\n', encoding='utf-8') as file:
102            for res in self.results:
103                file.write(res)
104                print(res)
105
106    def zipFile(self) -> None:
107        ''' Method zips results. '''
108        with ZipFile(f'{self.path}results.zip', 'w', compression=ZIP_DEFLATED) as file:
109            file.write(f'{self.path}results.txt')
110            print('Zip file info:')
111            for item in file.infolist():
112                print(f'Filename: {item.filename}')
113                print(f'''Last change: {datetime(item.date_time[0], item.date_time[1], item.date_time[2],
114                                              item.date_time[3], item.date_time[4], item.date_time[5])}''')
115                print(f'Compress type: {item.compress_type}')
116                print(f'Compress size: {item.compress_size}')
117                print(f'File size: {item.file_size}')
118
119

```

## Пример использования:

### Исходный текст:

```
1 When I have free time, I often listen to my favourite composer, Wolfgang Amadeus Mozart. His works fill me with new  
energy, help me to relax and raise my mood. He was a very talented musician. His father taught him the violin ;--),  
piano and musical theory. Little Mozart began to write music at the age of four. He wrote his first opera when he was  
eleven. When Mozart became an adult, he moved to Vienna. He had been successful in this town as a :))) child prodigy,  
but as an adult he found it difficult to find a job.  
2 In Vienna he met Haydn, who became his second father. Haydn supported the young composer and helped him in his musical  
career. Mozart's operas became very popular in the city. He did not spend much time thinking about his next  
composition. Musical ideas sprang from his mind and he just had ;-[[[[ to write them down. At this time he married  
Constance Weber and wrote one of his most famous works - «C minor» composition qrst123.  
3 Mozart enjoyed a successful career. He worked a lot. He visited Prague with his operas. Writing his last work  
'Requiem', commissioned by an unusual stranger, that it was his own requiem. He did not manage to finish his work and  
died at the age of 35 from poor health. The "Requiem" was completed by one of his pupils, Süssmayr. Mozart's 49  
symphonies and 18 operas are now world famous and are considered to have healing power.  
4
```

### Результаты анализа:

```
1 Sentenses amount: 21  
2 *****  
3 Declarative sentenses amount: 20  
4 Interrogative sentences amount: 0  
5 Incentive sentenses amount: 0  
6 *****  
7 Average sentence length: 51.095  
8 *****  
9 Average word length: 4.292  
10 *****  
11 Amount of smiles: 3  
12 *****  
13 Words with letters from 'a' to 'o' and digits:  
14 3lped  
15 m0  
16 *****  
17 String 12345 is not valid  
18 *****  
19 Amount of words in quotes: 3  
20 *****  
21 Amount of each letter:  
22 w - 29  
23 h - 64  
24 e - 121  
25 n - 69  
26 i - 83  
27 a - 88  
28 v - 11  
29 f - 24  
30 r - 62  
31 t - 78  
32 m - 50  
33 o - 83  
34 l - 30  
35 s - 72  
36 y - 16  
37 u - 35  
38 c - 29  
39 p - 25  
40 g - 16  
41 d - 45  
42 z - 6  
43 k - 6  
44 x - 2  
45 b - 11  
46 j - 3  
47 q - 4  
48 *****
```

```

Phrases in commas:
commissioned by an unusual stranger
his works fill me with new energy
i often listen to my favourite composer
*****
Zip file info:
Filename: task2/results.txt
Last change: 2024-04-11 20:02:22
Compress type: 8
Compress size: 390
File size: 1455
Do you want to finish task? (0 -> No | anything else -> Yes): yes
Task exit
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 0
Program finished!
PS D:\IGI\253502_KRASOV_11\IGI\LR4>

```

**Задание 3.** В соответствии с заданием своего варианта составить программу для вычисления значения функции с помощью разложения функции в степенной ряд. Задать точность вычислений eps. Предусмотреть максимальное количество итераций, равное 500.

$$\ln((x + 1)/(x - 1)) = 2 * \sum_{n=0}^{\infty} 1/((2n + 1) * x^{2n+1}), |x| > 1$$

Используя класс и обеспечить:

а) определение дополнительных параметров среднее арифметическое элементов последовательности, медиана, мода, дисперсия, СКО последовательности;

б) с помощью библиотеки matplotlib нарисовать графики разных цветов в одной координатной оси:

- график по полученным данным разложения функции в ряд, представленным в таблице,
- график соответствующей функции, представленной с помощью модуля math.
- Обеспечить отображение координатных осей, легенды, текста и аннотации.

в) сохранить графики в файл

Функция, запускающая задание:

```

32     @RunTask(Task3)
33     def runTask3(self, task: Task3) -> str | None:
34         ''' Task3\nWith a given accuracy calculate the value of a function using Tailor's row. Determine additional
35             parameters: arithmetic mean, median, mode, dispersion, standard deviation of the sequence. Using the
36             `matplotlib` library, draw charts of different colors on the same coordinate axis'''
37         return task()

```

Класс, вычисляющий дополнительный параметры последовательности:

```
1 from collections import Counter
2 from math import sqrt
3
4
5 class RowAnaliser():
6     def __init__(self) -> None:
7         ''' Inits class object. '''
8         self.row: list[float] = list()
9
10    def __call__(self) -> tuple[float, float, float, float]:
11        ''' Runs class methods. '''
12        if len(self.row) == 0:
13            return 'Tailor\'s row is empty!'
14        return (self.mean(), self.median(), self.mode(), self.variance(), self.stdev())
15
16    def mean(self) -> float:
17        ''' Method counts arithmetic mean of sequence. '''
18        return round(sum(self.row) / len(self.row), 4) if len(self.row) > 0 else 'Tailor\'s row is empty!'
19
20    def median(self) -> float:
21        ''' Method returns sequence median. '''
22        if len(self.row) == 0:
23            return 'Tailor\'s row is empty!'
24        sorted_row = sorted(self.row)
25        l = len(sorted_row)
26        return round(0.5*(sorted_row[l // 2 - 1] + sorted_row[l // 2]), 4)
27
28    def mode(self) -> str:
29        ''' Method returns sequence mode. '''
30        if len(self.row) == 0:
31            return 'Tailor\'s row is empty!'
32        value_counts = Counter(self.row)
33        max_count = max(value_counts.values())
34        return str([round(value, 4) for value,
35                   count in value_counts.items() if count == max_count])
36
37    def variance(self) -> float:
38        ''' Method counts sequence variance. '''
39        if len(self.row) == 0:
40            return 'Tailor\'s row is empty!'
41        elif len(self.row) < 2:
42            return 'Can\'t find variance. 2 and more elements needed.'
43        mean = self.mean()
44        return round(sum((s - mean) ** 2 for s in self.row) / len(self.row), 4)
45
46    def stdev(self) -> float:
47        ''' Method counts standart deviance of sequence. '''
48        if len(self.row) == 0:
49            return 'Tailor\'s row is empty!'
50        elif len(self.row) < 2:
51            return 'Can\'t find standart deviance. 2 and more elements needed.'
52        return round(sqrt(self.variance()), 4)
```

Класс, вычисляющий значение функции и сумму ряда Тейлора в точке:

```
1 from math import log
2 from task3.rowAnaliser import RowAnaliser
3
4
5 class TailorSum(RowAnaliser):
6     def __init__(self) -> None:
7         ''' Inits class object. '''
8         super().__init__()
9
10    def __call__(self):
11        ''' Clears list with row elements. '''
12        self.row.clear()
13
14    def countMath(self, x: float):
15        ''' Method returns function value in the x point. '''
16        return log((x + 1)/(x - 1))
17
18    def countTailor(self, x: float, eps: float, mathResult: float) -> tuple[float, int]:
19        ''' Function that counts sum of Tailor's row with the required accuracy. '''
20        tailorSum = 0
21        N = -1
22        while N + 1 < 500:
23            N += 1
24            element = 1/((2*N+1)*x***(2*N+1))
25            tailorSum += element
26            if (abs(2*tailorSum - mathResult) >= eps/10):
27                continue
28            break
29        self.row.append(2*tailorSum)
30        return 2*tailorSum, N+1
31
32    def getAdditionalParams(self) -> str:
33        ''' Function returns additional parameters of sequence. '''
34        params = super().__call__()
35        if type(params) == str:
36            return params
37        print(f'Mode: {params[2]}\n')
38        return f'Additional parameters:\nArithmetics mean: {params[0]}\n\
39 Median: {params[1]}\nVariance: {params[3]}\nStandard deviation: {params[4]}'
40
```

### Общий класс задания 3:

```
1 from utils import validateNumberInput
2 import matplotlib.pyplot as plt
3 from statistics import median
4 from task3.tailorSum import TailorSum
5
6
7 class Task3():
8     def __init__(self) -> None:
9         ''' Inits class object. '''
10        self.tailor = TailorSum()
11        self.mathResults = list()
12        self.tailorResults = list()
13        self.iterations = list()
14
15    def __call__(self) -> None | str:
16        ''' Runs class methods. '''
17        self.validateInput()
18        if self.eps == 0:
19            return 'stop'
20        self.tailor()
21        self.displayConsole()
22        self.displayCharts()
23
24    def validateInput(self) -> None:
25        ''' Method for validation input of accuracy and x value. '''
26        self.eps = validateNumberInput(
27            'accuracy (0 < acc < 1) (0 -> Exit function)', '0 < acc < 1', float, condition=lambda x: x < 0 or x >= 1)
28        if self.eps == 0:
29            return
30        start = validateNumberInput(
31            'start x (|x| > 1)', '|x| must be bigger than 1', float, condition=lambda x: abs(x) <= 1)
32        while True:
33            end = validateNumberInput(
34                'end x (|x| > 1 & x > start)', '|x| must be bigger than 1', float, condition=lambda x: abs(x) <= 1)
35            if end <= start:
36                print('Wrong input -> (end point > start point)! Try again!\n')
37                continue
38            break
39        N = validateNumberInput(
40            'number of points (N > 0)', 'N - integer positive number', int, condition=lambda x: x <= 0)
41        h = (end - start) / N
42        self.x = [round(start + i*h, 4) for i in range(N + 1)]
```

```

44     def printBorders(self, type: str, length: int) -> None:
45         ''' Method prints result table borders. '''
46         for i in range(5):
47             if i == 0:
48                 print("—" + "—" * length,
49                      end="—" if type == 'top' else print("\n|—" + "—" * length, end="|—"))
50                 if type == 'mid' else print("\n|—" + "—" * length, end="|—"))
51             elif i != 4:
52                 print("—" * (length + 3), end="—" if type == 'top' else print("—" * (length + 3), end="|—"))
53                     if type == 'mid' else print("—" * (length + 3), end="|—"))
54             else:
55                 print("—" * (length + 3), end="—" + "\n") if type == 'top' else print("—" * (length + 3), end="|—" + "\n")
56                     if type == 'mid' else print("—" * (length + 3), end="|—" + "\n"))
57
58     def displayConsole(self) -> None:
59         ''' Method displays the result in console. '''
60         for x in self.x:
61             self.mathResults.append(self.tailor.countMath(x))
62             tailorSum, iterations = self.tailor.countTailor(
63                 x, self.eps, self.mathResults[-1])
64             self.tailorResults.append(tailorSum)
65             self.iterations.append(iterations)
66             length = max(len(str(max(self.tailorResults))), len(str(max(self.iterations))),
67                           len(str(self.eps)), len(str(max(self.x))), 9)
68             data = [[x, n, F(x), "Math F(x)", "eps"]]
69             for i in range(len(self.x)):
70                 data.append([self.x[i], self.iterations[i], round(self.tailorResults[i], len(str(self.eps))),
71                             round(self.mathResults[i], len(str(self.eps))), self.eps])
72             self.printBorders('top', length)
73             for i in range(len(self.x) + 1):
74                 for j in range(5):
75                     if j == 0:
76                         print("—" + str(data[i][j]) + "—" *
77                               (length - len(str(data[i][j]))), end="—" + " ")
78                     else:
79                         print(str(data[i][j]).format() + "—" * (length + 3 -
80                               len(str(data[i][j]))), end="—" + " ")
81             self.printBorders(
82                 'mid', length) if i != len(self.x) else self.printBorders('bot', length)
83         print()

```

```

85     def displayCharts(self):
86         ''' Method creates function and row charts. '''
87         plt.figure()
88         plt.plot(self.x, self.mathResults, label='Math function')
89         plt.plot(self.x, self.tailorResults, label='Tailor\'s row')
90         plt.annotate(self.tailor.getAdditionalParams(),
91                     (median(self.x) + 0.1, median(self.tailorResults) + 0.1))
92         plt.legend()
93         plt.title('Function\'s and Tailor\'s row charts')
94         plt.xlabel('x')
95         plt.ylabel('y')
96         plt.savefig('./task3/charts.png')
97

```

## Пример использования:

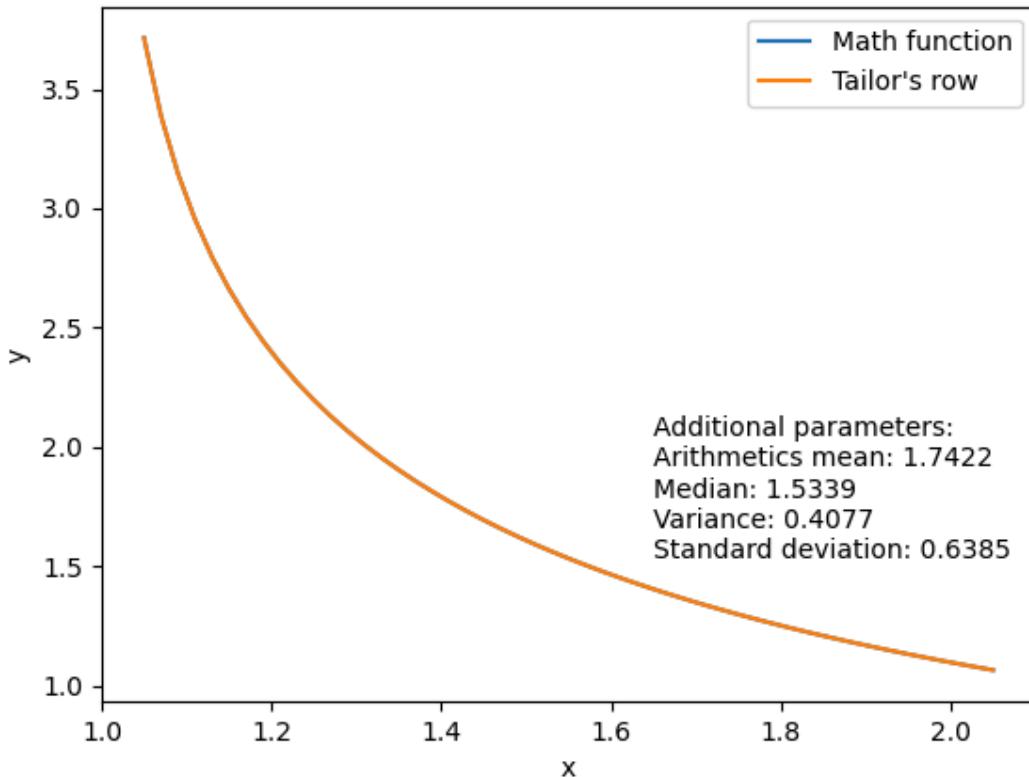
```
PS D:\IGI\253502_KRASYOV_11\IGI\LR4> python main.py
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 3
Task3
With a given accuracy calculate the value of a function using Tailor's row. Determine additional parameters: arithmetic mean, median, mode, dispersion, standard deviation of the sequence. Using the matplotlib library, draw charts of different colors on the same coordinate axis
Enter accuracy (0 < acc < 1) (0 -> Exit function): 1e-6
Enter start x (|x| > 1): 1.05
Enter end x (|x| > 1 & x > start): 2.05
Enter number of points (N > 0): 50
```

| x    | n   | F(x)    | Math F(x) | eps   |
|------|-----|---------|-----------|-------|
| 1.05 | 138 | 3.71357 | 3.71357   | 1e-06 |
| 1.07 | 100 | 3.38681 | 3.38681   | 1e-06 |
| 1.09 | 79  | 3.14511 | 3.14511   | 1e-06 |
| 1.11 | 65  | 2.95396 | 2.95396   | 1e-06 |
| 1.13 | 56  | 2.79634 | 2.79634   | 1e-06 |

|      |    |         |         |       |
|------|----|---------|---------|-------|
| 1.99 | 10 | 1.10532 | 1.10532 | 1e-06 |
| 2.01 | 10 | 1.09199 | 1.09199 | 1e-06 |
| 2.03 | 10 | 1.079   | 1.079   | 1e-06 |
| 2.05 | 10 | 1.06635 | 1.06635 | 1e-06 |

```
Mode: [3.7136, 3.3868, 3.1451, 2.954, 2.7963, 2.6626, 2.5467, 2.4446, 2.3536, 2.2717, 2.1972, 2.1291, 2.0664, 2.0084, 1.9545, 1.9042, 1.8571, 1.8129, 1.7712, 1.7319, 1.6946, 1.6592, 1.6256, 1.5936, 1.5631, 1.5339, 1.506, 1.4793, 1.4536, 1.429, 1.4053, 1.3826, 1.3606, 1.3394, 1.319, 1.2993, 1.2802, 1.2618, 1.2439, 1.2266, 1.2098, 1.1936, 1.1778, 1.1625, 1.1476, 1.1331, 1.119, 1.1053, 1.092, 1.079, 1.0664]
```

Function's and Tailor's row charts



**Задание 4.** В соответствии с заданием своего варианта разработать базовые классы и классы наследники. Построить квадрат, на одной стороне которого, как на основании, построен равносторонний треугольник со стороной a.

Функция, запускающая задание

```
1 @RunTask(Task4)
2 def runTask4(self, task: Task4) -> str | None:
3     """ Task4\nConstruct a square, on one side of which, as a base, an equilateral triangle with side a is built.
4     ...
5
6     return task()
```

Абстрактный класс `Геометрическая фигура`:

```
1 from abc import ABC
2
3
4 class GeometricFigure(ABC):
5     def area(self):
6         """ Method returns area of the figure. """
7         pass
8
```

Класс `Цвет фигуры`:

```
1 class FigureColor():
2     def __init__(self, color: str) -> None:
3         self._color = color
4
5     @property
6     def color(self) -> str:
7         """ Square name getter. """
8         return self._color
9
```

Класс-примесь для классов квадрата и треугольника:

```
1 class FigureMixin(object):
2     def __str__(self) -> str:
3         """ Method returns figure info. """
4         return 'Name: {}\nColor: {}\nSide: {}\nArea: {}'.format(self.name, self.color.color, self.side, self.area())
5
```

Класс квадрата:

```
1 from task4.geometricFigure import GeometricFigure
2 from task4.figureColor import FigureColor
3 from task4.figureMixin import FigureMixin
4
5
6 class Square(GeometricFigure, FigureMixin):
7     def __init__(self, side: float, color: str) -> None:
8         ''' Inits class object. '''
9         self.side = side
10        self.color = FigureColor(color)
11
12    def area(self) -> float:
13        ''' Method returns square area. '''
14        return self.side**2
15
16    @property
17    def name(self) -> str:
18        ''' Square name getter. '''
19        return self._name
20
21    @name.setter
22    def name(self, value: str) -> None:
23        ''' Square name setter. '''
24        self._name = value.capitalize()
25
```

Класс треугольника:

```
1 from task4.geometricFigure import GeometricFigure
2 from task4.figureColor import FigureColor
3 from task4.figureMixin import FigureMixin
4 from math import sqrt
5
6
7 class Triangle(GeometricFigure, FigureMixin):
8     def __init__(self, side: float, color: str) -> None:
9         ''' Inits class object. '''
10        self.side = side
11        self.color = FigureColor(color)
12
13    def area(self) -> float:
14        ''' Method returns triangle area. '''
15        return self.side**2 * sqrt(3)/4
16
17    @property
18    def name(self) -> str:
19        ''' Triangle name getter. '''
20        return self._name
21
22    @name.setter
23    def name(self, value: str) -> None:
24        ''' Triangle name setter. '''
25        self._name = value.capitalize()
26
```

Общий класс для задания 4:

```
1 from utils import validateNumberInput, validateColorInput
2 from task4.square import Square
3 from task4.triangle import Triangle, sqrt
4 from matplotlib import pyplot as plt
5
6
7 class Task4():
8     def __init__(self) -> None:
9         ''' Inits class object. '''
10        self.square = None
11        self.triangle = None
12
13    def __call__(self) -> None | str:
14        ''' Runs class methods. '''
15        self.validateInput()
16        if self.square is None:
17            return 'stop'
18        self.displayInfo()
19        self.displayCharts()
20
21    def validateInput(self) -> None:
22        ''' Function for validation input of square side. '''
23        side = validateNumberInput('length of square side (0 -> Exit function)', 'length must be bigger 0', float,
24                                  condition=lambda x: x < 0)
25        if side == 0:
26            self.square = None
27            return
28        self.square = Square(side, validateColorInput('square'))
29        self.triangle = Triangle(side, validateColorInput('triangle'))
30        self.square.name = input('Enter square name: ')
31        self.triangle.name = input('Enter triangle name: ')
32
33    def displayInfo(self) -> None:
34        ''' Method displays figures info in console. '''
35        print('\nSquare info:')
36        print(self.square())
37        print('\nTriangle info:')
38        print(self.triangle())
```

```

40     def displayCharts(self) -> None:
41         ''' Method displays figures charts. '''
42         side = self.square.side
43         square_x = [0, side, side, 0, 0]
44         square_y = [0, 0, side, side, 0]
45         triangle_x = [side, side, (sqrt(3)*0.5 + 1)*side, side]
46         triangle_y = [0, side, 0.5*side, 0]
47         plt.figure()
48         plt.xlim(0, (sqrt(3)*0.5 + 1)*side + 1)
49         plt.ylim(0, side + 1)
50         plt.axis('equal')
51         plt.plot(square_x, square_y, color=self.square.color.color, marker='o',
52                  markersize=5, linestyle='-')
53         plt.plot(triangle_x, triangle_y, color=self.triangle.color.color, marker='o',
54                  markersize=5, linestyle='-', label=self.triangle.name)
55         plt.annotate(self.square.name, (side/2 -
56                                         len(self.square.name)*0.1, side/2))
57         plt.annotate(self.triangle.name,
58                     ((sqrt(3)*0.25 + 1)*side -
59                      len(self.triangle.name)*0.1, side/2))
60         plt.fill(square_x, square_y, color=self.square.color.color, alpha=0.6)
61         plt.fill(triangle_x, triangle_y,
62                  color=self.triangle.color.color, alpha=0.6)
63         plt.autoscale()
64         plt.savefig('./task4/charts.png')
65

```

Пример использования:

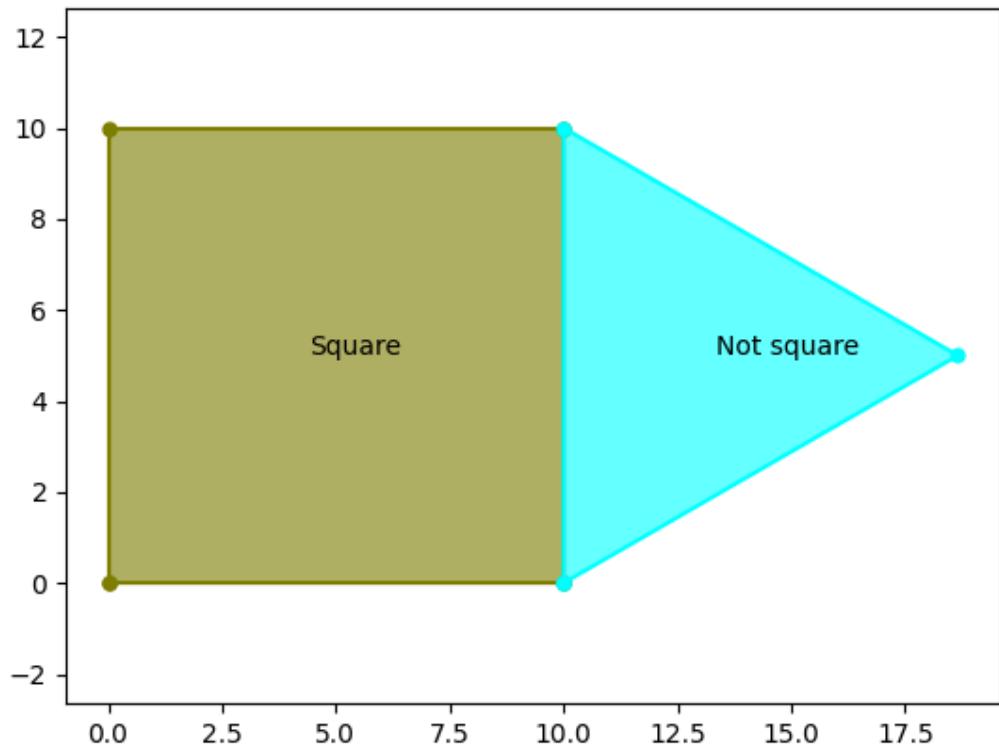
```

PS D:\IGI\253502_KRASOV_11\IGI\LR4> python main.py
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 4
Task4
Construct a square, on one side of which, as a base, an equilateral triangle with side a is built.
Enter length of square side (0 -> Exit function): 10
Input square color: olive
Input triangle color: cyan
Enter square name: Square
Enter triangle name: Not square

Square info:
Name: Square
Color: olive
Side: 10.0
Area: 100.0

Triangle info:
Name: Not square
Color: cyan
Side: 10.0
Area: 43.30127018922193
Do you want to finish task? (0 -> No | anything else -> Yes): yes
Task exit
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 0
Program finished!
PS D:\IGI\253502_KRASOV_11\IGI\LR4>

```



**Задание 5.** В соответствии с заданием своего варианта исследовать возможности библиотека NumPy при работе с массивами и математическими и статическими операциями. Сформировать целочисленную матрицу  $A[n,m]$  с помощью генератора случайных чисел (random). В соответствии с заданием своего варианта исследовать возможности библиотека NumPy при работе с массивами и математическими и статическими операциями. Сформировать целочисленную матрицу  $A[n,m]$  с помощью генератора случайных чисел (random).

Функция, запускающая задание:

```

42     @RunTask(Task5)
43     def runTask5(self, task: Task5) -> str | None:
44         """ Task5\nGenerate an integer matrix A[n,m] using a random number generator. Find all elements that exceed the
45             specified number B in absolute value. Count the number of such elements and write them to array C. Calculate
46             the median value for this array C. """
47         return task()

```

Класс, хранящий матрицу и анализирующий её:

```
1 from generators import matrixGenerator
2 import numpy as np
3
4
5 class Matrix():
6     def __call__(self, rows: int, columns: int) -> None:
7         ''' Method generates matrix[rows, columns]. '''
8         self.matrix = np.array(matrixGenerator(rows, columns))
9         print('Generated matrix:')
10        for row in self.matrix:
11            print(row)
12        print()
13
14    def findElements(self, number: float):
15        ''' Method returns numpy array with elements greater than `number`. '''
16        return self.matrix[np.abs(self.matrix) > number]
17
18    def findMedianNp(self, array):
19        ''' Method counts array median. '''
20        return np.median(array)
21
22    def findMedian(self, array) -> float:
23        ''' Method counts array median. '''
24        array.sort()
25        l = len(array)
26        return array[l // 2] if l % 2 else (array[l // 2 - 1] + array[l // 2]) / 2
27
```

Общий класс для задания 5:

```
1 from utils import validateNumberInput
2 from task5.matrix import Matrix
3
4
5 class Task5(Matrix):
6     def __init__(self) -> None:
7         ''' Inits class object. '''
8         self.matrix = None
9
10    def __call__(self) -> None | str:
11        ''' Runs class methods. '''
12        self.validateInput()
13        if self.matrix is None:
14            return 'stop'
15        self.analiseMatrix()
16
17    def validateInput(self) -> None:
18        ''' Method validates number of matrix rows & columns input. '''
19        rows = validateNumberInput('number of matrix rows (0 -> Exit function)',
20                                    'number must be bigger 0', int, condition=lambda x: x < 0)
21        if rows == 0:
22            self.matrix = None
23            return
24        columns = validateNumberInput('number of matrix columns',
25                                    'number must be bigger 0', int, condition=lambda x: x <= 0)
26        super().__call__(rows, columns)
27
28    def printArray(self, array, number: float):
29        ''' Method returns numpy array of elements greater than `number`.'''
30        print(f'Amount of elements greater than |{number}|: {len(array)}')
31        if len(array) > 0:
32            print('Elements:')
33            for el in array:
34                print(el, end=" ")
35            print()
36
37    def analiseMatrix(self):
38        ''' Method analises matrix. '''
39        number = validateNumberInput('any number', '', float)
40        C = self.findElements(number)
41        self.printArray(C, number)
42        if len(C) == 0:
43            print('Array `C` is empty! Can\'t find median!')
44            return
45        print(f'NumPy median -> {self.findMedianNp(C)}')
46        print(f'Programmed median -> {self.findMedian(C)}')
47
```

## Пример использования:

```
PS D:\IGI\253502_KRASOV_11\IGI\LR4> python main.py
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 5
Task5
Generate an integer matrix A[n,m] using a random number generator.Find all elements that exceed the specified number B in absolute value. Count the number of such elements and write them to array C. Calculate the median value for this array C.
Enter number of matrix rows (0 -> Exit function): 3
Enter number of matrix columns: 4
Generated matrix:
[ 41  79 -145 138]
[ 57 127 125  37]
[ 44  19 -16 102]

Enter any number: 75
Amount of elements greater than 75.0: 6
Elements:
79 -145 138 127 125 102
NumPy median -> 113.5
Programmed median -> 113.5
Do you want to finish task? (0 -> No | anything else -> Yes): нуы
Task exit
Enter number of task from 1 to 5, or 6 to perform data analysis, or 0 to exit: 0
Program finished!
PS D:\IGI\253502_KRASOV_11\IGI\LR4>
```

**Дополнительное задание.** Используя любой из наборов данных (datasets) на ресурсе: <https://www.kaggle.com/datasets> исследовать основные возможности библиотеки Pandas.

Датасет - модели машин с основными характеристиками и ценой.

Класс для анализа данных:

```
1 import pandas as pd
2 from sys import maxsize
3 from analysis.plotDrawer import PlotDrawer
4
5
6 class DataAnalisis(PlotDrawer):
7     def __init__(self, path: str = './data.csv') -> None:
8         """ Inits class object. """
9         self.df = pd.read_csv(path)
10
11    def __call__(self) -> None:
12        """ Method runs class methods. """
13        self.getInfo()
14        print('Dataset analisys:\n')
15        self.getYearDifference()
16        self.getMileageDifference()
17        self.getAvgMpg()
18        self.drawYearPricePlot(self.df)
19        self.drawMileagePricePlot(self.df)
20        self.drawMpgPricePlot(self.df)
21
22    def getInfo(self) -> None:
23        """ Method returns dataset info. """
24        print('Dataset:')
25        print(self.df.info())
26        print(self.df)
27
28    def getAllModels(self, model: str, year: int = 1970, price: int = maxsize, fuelType: str = 'Petrol') -> None:
29        """ Method returns all cars satisfying release year ('year'), max price ('price') & fuel type ('fuelType'). """
30        cars = self.df[(self.df['model'] == model) & (self.df['year'] >= year) &
31                      (self.df['price'] <= price) & (self.df['fuelType'] == fuelType)]
32        print(cars)
33
34    def getAvgPrice(self, yearStart: int = 1970, yearEnd: int = 2024) -> float:
35        """ Method returns price mean of cars released between 'yearStart' & 'yearEnd'. """
36        cars = self.df[(self.df['year'] <= yearEnd) &
37                      (self.df['year'] >= yearStart)]
38        return cars['price'].mean()
39
40    def getMedianTax(self) -> float:
41        """ Method returns median car tax. """
42        return self.df['tax'].median()
```

```

44     def getAvgMpg(self) -> None:
45         ''' Method compares car's MPG depending on tax. '''
46         print('MPG compare depending on tax:')
47         tax = self.getMedianTax()
48         print(f'Average tax: {tax}')
49         print(
50             f'Average MPG of cars with tax >= {tax}: {self.df[self.df["tax"] >= tax]["mpg"].mean():.2f}\n')
51         print(
52             f'Average MPG of cars with tax < {tax}: {self.df[self.df["tax"] < tax]["mpg"].mean():.2f}\n')
53
54     def getYearDifference(self, year: int = 2019):
55         ''' Method compares car's price depending on release year. '''
56         print('Price compare depending on release year:')
57         p1, p2 = self.getAvgPrice(
58             yearEnd=year), self.getAvgPrice(yearStart=year)
59         print(f'Average price before {year}: ${p1:.0f}')
60         print(f'Average price after {year}: ${p2:.0f}')
61         print(f'Ratio (after {year} / before {year}): {p2/p1:.2f}\n')
62
63     def getMileageDifference(self):
64         ''' Method compares car's price depending on mileage'''
65         print('Price compare depending on car mileage:')
66         maxMil = self.df.iloc[self.df['mileage'].idxmax()]
67         print('Max mileage car:')
68         self.printModelInfo(maxMil)
69         minMil = self.df.iloc[self.df['mileage'].idxmin()]
70         print('Min mileage car:')
71         self.printModelInfo(minMil)
72         print(
73             f'Price difference: {minMil["price"]} - {maxMil["price"]} = {minMil["price"] - maxMil["price"]}\n')
74
75     def printModelInfo(self, car) -> None:
76         ''' Method returns car info. '''
77         print(f'Model: {car["model"]}')
78         print(f'Year: {car["year"]}')
79         print(f'Price: {car["price"]}')
80         print(f'Transmission: {car["transmission"]}')
81         print(f'Mileage: {car["mileage"]}')
82         print(f'FuelType: {car["fuelType"]}')
83         print(f'Tax: {car["tax"]}')
84         print(f'MPG: {car["mpg"]}')
85         print(f'EngineSize: {car["engineSize"]}\n')
86         print(f'Manufacturer: {car["Manufacturer"]}\n')
87

```

Класс, выполняющий построение графиков по данным:

```
1 from matplotlib import pyplot as plt
2
3
4 class PlotDrawer():
5     def drawYearPricePlot(self, df):
6         ''' Method draws average car price chart depending on year. '''
7         years, prices = [], []
8         for year in range(1995, 2025):
9             cars = df[df['year'] == year]
10            if len(cars) > 0:
11                years.append(year)
12                prices.append(cars['price'].mean())
13
14            plt.figure()
15            plt.title('Average car price 1995-2024')
16            plt.plot(years, prices, marker='o', markersize=2, linestyle='--')
17            plt.xlabel('Years')
18            plt.ylabel('Average price')
19            plt.savefig('./analysis/year-price.png')
20
21        def drawMileagePricePlot(self, df):
22            ''' Method draws average car price chart depending on mileage '''
23            mileages, prices = [], []
24            for m in range(1, 41):
25                cars = df[(df['mileage'] < int(
26                    1e4*m)) & (df['mileage'] > int(1e4*(m - 1)))]
27                if (len(cars) == 0):
28                    continue
29                mileages.append(1e4*m)
30                prices.append(cars['price'].mean())
31
32            plt.figure()
33            plt.title('Average car price depending on mileage')
34            plt.plot(mileages, prices, marker='o', markersize=2, linestyle='--')
35            plt.xlabel('Mileages')
36            plt.ylabel('Average price')
37            plt.savefig('./analysis/mileage-price.png')
38
39        def drawMpgPricePlot(self, df):
40            ''' Method draws average car price chart depending on mpg '''
41            mpes, prices = [], []
42            for i in range(10, 480, 10):
43                cars = df[(df['mpg'] < i) & (df['mpg'] > i - 10)]
44                if len(cars) == 0:
45                    continue
46                mpes.append(i)
47                prices.append(cars['price'].mean())
48
49            plt.figure()
50            plt.title('Average car price depending on mpg')
51            plt.plot(mpes, prices, marker='o', markersize=2, linestyle='--')
52            plt.xlabel('MPG')
53            plt.ylabel('Average price')
54            plt.savefig('./analysis/mpg-price.png')
```

## Пример использования:

```
[1] > from analysis import DataAnalysys
  ds = DataAnalysys()
  ds.df.head()
  ds.df

...      model  year  price  transmission  mileage  fuelType  tax  mpg  engineSize  Manufacturer
0     I10  2017   7495       Manual    11630  Petrol  145  60.1        1.0      hyundai
1     Polo  2017  10989       Manual     9200  Petrol  145  58.9        1.0    volkswagen
2   2 Series  2019  27990  Semi-Auto    1614  Diesel  145  49.6        2.0        BMW
3   Yeti Outdoor  2017  12495       Manual    30960  Diesel  150  62.8        2.0       skoda
4    Fiesta  2017   7999       Manual    19353  Petrol  125  54.3        1.2        ford
...
...      model  year  price  transmission  mileage  fuelType  tax  mpg  engineSize  Manufacturer
97707  Fiesta  2017  10447  Automatic    8337  Petrol  145  54.3        1.0        ford
97708  3 Series  2014  14995       Manual    25372  Diesel  30  61.4        2.0        BMW
97709  Fiesta  2017   8950       Manual    19910  Petrol  125  54.3        1.2        ford
97710   Astra  2017  10700  Automatic   24468  Petrol  125  50.4        1.4  vauxhall
97711 Grandland X  2019  15798       Manual    10586  Diesel  150  48.7        1.5  vauxhall
97712 rows × 10 columns

[2] > ds.df.info()

... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 97712 entries, 0 to 97711
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype  
---  -- 
 0   model       97712 non-null   object 
 1   year        97712 non-null   int64  
 2   price       97712 non-null   int64  
 3   transmission 97712 non-null   object 
 4   mileage     97712 non-null   int64  
 5   fuelType    97712 non-null   object 
 6   tax          97712 non-null   int64  
 7   mpg          97712 non-null   float64
 8   engineSize  97712 non-null   float64
 9   Manufacturer 97712 non-null   object 
dtypes: float64(2), int64(4), object(4)
memory usage: 7.5+ MB

[3] > cars = ds.df[(ds.df['model'] == 'A4') & (ds.df['year'] >= 2017) &
  ...           | (ds.df['price'] <= 20000)]
  ds.df.head()
  cars

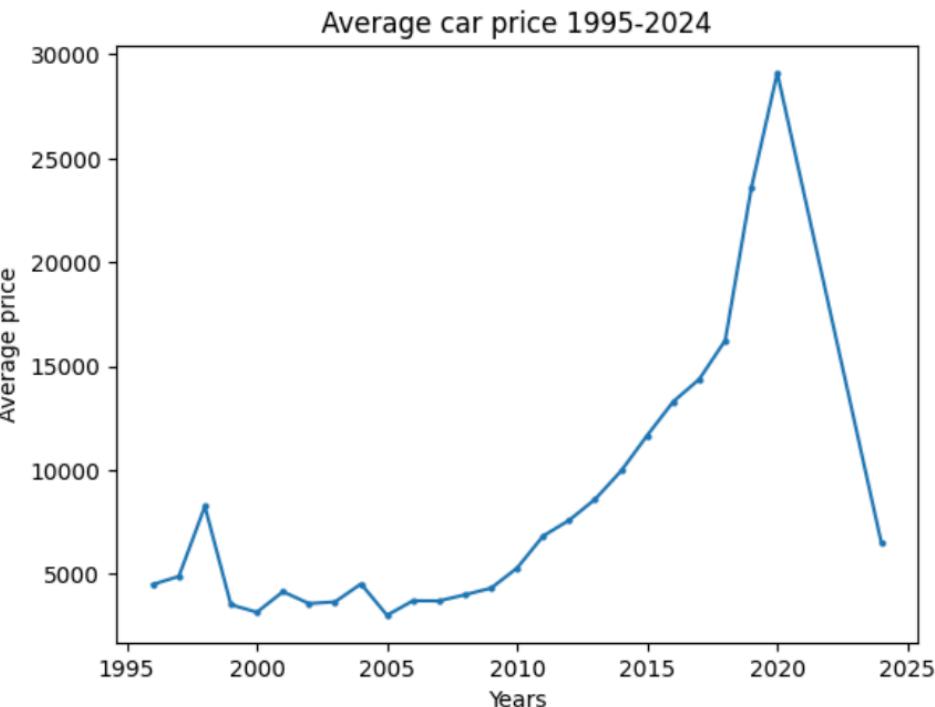
...      model  year  price  transmission  mileage  fuelType  tax  mpg  engineSize  Manufacturer
28     A4  2017  18182       Manual    18574  Petrol  145  51.4        1.4        Audi
569    A4  2017  15540  Automatic    35452  Diesel  145  70.6        2.0        Audi
611    A4  2017  18990  Semi-Auto    30417  Diesel  30  62.8        2.0        Audi
672    A4  2018  18995  Semi-Auto    18696  Petrol  145  50.4        1.4        Audi
1108   A4  2017  17190       Manual    35790  Diesel  30  67.3        2.0        Audi
...
...      model  year  price  transmission  mileage  fuelType  tax  mpg  engineSize  Manufacturer
96842   A4  2017  15441       Manual    49000  Diesel  30  64.2        2.0        Audi
97030   A4  2017  17445       Manual    36187  Diesel  20  70.6        2.0        Audi
97313   A4  2017  17490       Manual    35000  Petrol  125  49.6        2.0        Audi
97393   A4  2018  17495  Semi-Auto    9296  Petrol  150  53.3        1.4        Audi
97395   A4  2017  20000  Automatic   20852  Diesel  30  65.7        2.0        Audi
306 rows × 10 columns

[4] > ds.getYearDifference()

... Price compare depending on release year:
Average price before 2019: $16,242
Average price after 2019: $24,310
Ratio (after 2019 / before 2019): 1.50
```

```
ds.drawYearPricePlot(ds.df)
```

[5]



```
ds.getMileageDifference()
```

[6]

... Price compare depending on car mileage:

Max mileage car:

Model: A6  
Year: 2008  
Price: 2490  
Transmission: Manual  
Mileage: 323000  
FuelType: Diesel  
Tax: 200  
MPG: 44.1  
EngineSize: 2.0  
Manufacturer: Audi

Min mileage car:

Model: C Class  
Year: 2019  
Price: 26995  
Transmission: Semi-Auto  
Mileage: 1  
FuelType: Diesel  
Tax: 145  
MPG: 54.3  
EngineSize: 1.6  
Manufacturer: merc

Price difference: 26995 - 2490 = 24505

