Лабораторная работа №4
по дисциплине ИГИ
Вариант 27

Выполнил: Шумский Д. С., гр. 253501

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

27. Приняв способ изображения рационального числа в виде записи с двумя полями [числитель, знаменатель] целого типа написать программу, позволяющую: а) определить, есть ли среди 10 рациональных чисел равные; б) вычислить наибольшее из данных рациональных чисел (числа не обязательно имеют несократимую форму). Для хранения рациональных чисел использовать словарь. Выведите информацию о числе, введенном с клавиатуры

```python
numbers = {
    "one": Fraction(10000, 10000),
    "one_2": Fraction(76, 76),
    "minus_one_fiftieth": Fraction(-1, 50),
    "half": Fraction(1, 2),
    "two_thirds": Fraction(2, 3),
    "three_fourths": Fraction(3, 4),
    "five": Fraction(20, 4),
    "one_twentieth": Fraction(1, 20),
    "and_minus_one_fiftieth": Fraction(40, -2000),
    "a_half": Fraction(45, 90)
}


 Daniil Shumskiy *
def task_1():
    pickleSerializer = PickleSerializer("src/task_1/data/binary.bin")
    csvSerializer = CSVSerializer("src/task_1/data/data.csv")

    pickleSerializer.save(FracCollection(numbers))
    csvSerializer.save(pickleSerializer.obtain())
    fractions = csvSerializer.obtain()

    frac = Report.input_frac()

    Report.make(fractions, frac)
```

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.

27. Вывести все заглавные английские буквы. В заданном тексте заменить последовательность символов «p...pb...bc...c» (букв p и c в последовательности больше 0, букв b – больше единицы) на последовательность «ddd». определить количество слов, длина которых меньше 5; найти самое короткое слово, заканчивающееся на букву 'd'; вывести все слова в порядке убывания их длин.

```
import re


class TextReport:
    """Processing the text according to the specific rules"""

    def __init__(self, text='') -> None:
        self.__text = text

    @property
    def text(self):
        return self.__text

    @text.setter
    def text(self, input_text):
        self.__text = input_text

    def sentences_count(self):
        sentences = re.findall(r'[A-Z0-9][^?!.]*[!?.]', self.__text)
        return len(sentences)

    def decl_sent_count(self):
        sentences = re.findall(r'[A-Z0-9][^?!.]*\.', self.__text)
        return len(sentences)

    def excl_sent_count(self):
        sentences = re.findall(r'[A-Z0-9][^?!.]*!', self.__text)
        return len(sentences)

    def interrog_sent_count(self):
        sentences = re.findall(r'[A-Z0-9][^?!.]*\?', self.__text)
        return len(sentences)

    def avg_sent_size(self):
        sentences = re.findall(r'[A-Z0-9][^?!.]*[!?.]', self.__text)
        sizesum = 0
        for snt in sentences:
            sizesum += len(re.split(r"[,; ]", snt))
        return sizesum / len(sentences)

    def avg_word_size(self):
        wordCounter = 0
        sizeSum = 0
        for w in self.words_list():
            wordCounter += 1
```

```
            sizeSum += len(w)
        return sizeSum / wordCounter

    def emojis_count(self):
        emojis = re.findall(r'([:;]-+(\[+|]+|\(+|\)+))', self.__text)
        return len(emojis)

    def all_capital_symbols(self):
        return re.findall(r'[A-Z]', self.__text)

    @staticmethod
    def replace_sequence(text):
        return re.sub(r'p.*?pb.*?bc.*?c', 'ddd', text)

    def words_list(self):
        words_list = re.split(r"[,;.!?\n ]", self.__text)
        return [w for w in words_list if w != ""]

    def words_less_five(self):
        return sum(1 for w in self.words_list() if len(w) < 5)

    def find_smallest_word_ends_d(self):
        word = ''
        minWordSize = 1e9
        for w in self.words_list():
            if w.endswith('d') and len(w) < minWordSize:
                word = w
                minWordSize = len(word)
        return word

    def sort_words(self):
        return sorted(self.words_list(), key=lambda x: len(x),
reverse=True)
```

Задание 3.

В соответствии с заданием своего варианта доработать программу из ЛР3, использовав 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.
- Сохранить графики в файл

27. Функция asin

```python
import matplotlib.pyplot as plt
import math
import statistics

class PythonVsTaylor:
    """Class that printing information about statistics of taylor sequence"""

    @staticmethod
    def __save_plot(x, y_t, y_m, path='src/task_3/graph.png'):
        """Function for plot."""
        plt.plot(x, y_t, label='taylor asin')
        plt.plot(x, y_m, label='python asin')

        plt.xlabel("X")
        plt.ylabel("Y")
        plt.legend()

        plt.grid(visible=True)
        plt.axhline(y=0, color='k')
        plt.axvline(x=0, color='k')

        plt.savefig(path)

    @staticmethod
    def __asin_taylor(x, eps=0.1):
        sm = 0
        iter_count = 0
        arcsin_math = math.asin(x)

        while abs(sm - arcsin_math) > eps and iter_count < 500:
            sm += math.factorial(2 * iter_count) / (
                    4 ** iter_count * math.factorial(iter_count) ** 2 * (2 * iter_count + 1)) * x ** (2 * iter_count + 1)
            iter_count += 1

        return sm

    @classmethod
    def run(cls, taylor_eps=0.1):
        y_t = []
        y_m = []
        xs = []
        for x in map(lambda i: i / 100, range(-100, 100, 1)):
            xs.append(x)
            y_t.append(cls.__asin_taylor(x, taylor_eps))
            y_m.append(math.asin(x))

        print(f"Mean = {statistics.mean(y_t)}")
        print(f"Median = {statistics.median(y_t)}")
        print(f"Moda = {statistics.mode(y_t)}")
        print(f"Variance = {statistics.variance(y_t)}")
        print(f"Standard deviation = {statistics.stdev(y_t)}")

        cls.__save_plot(xs, y_t, y_m)
```

Задание 4.

В соответствии с заданием своего варианта разработать базовые классы и классы наследники.

Программа должна содержать следующие базовые функции:

1) ввод значений параметров пользователем;

2) проверка корректности вводимых данных;

3) построение, закрашивание фигуры в выбранный цвет, введенный с клавиатуры, и подпись фигуры текстом, введенным с клавиатуры;

4) вывод фигуры на экран и в файл.

27. Шестиугольник с известной стороной ***a***

```
import math
from math import sqrt, cos, sin
from PIL import Image, ImageDraw, ImageColor, ImageFont

from src.task_4.figure_colour import FigureColour
from src.task_4.geometric_figure import GeometricFigure
from src.task_4.name_mixin import NameMixin


class Hexagon(GeometricFigure, NameMixin):
    """Hexagon figure class"""
    def __init__(self, a: float, color: FigureColour, name: str):
        super().__init__(name)
        self.__a = a
        self.__color = color

    def calculateS(self):
        """Calculating value of the figure area """
        return 3 * sqrt(3) * self.__a ** 2 / 2

    def __str__(self):
        return "Colour: {colour}\nArea:
{area}".format(colour=self.__color.color, area=self.calculateS())

    def draw(self):
        """Drawing the figure with Pillow library"""
        points = []
        a = self.__a
        for i in range(0, 6):
            angle_deg = 60 * i
            angle_rad = math.pi / 180 * angle_deg
            points.append((500 + a * cos(angle_rad), 500 + a *
sin(angle_rad)))
        lowest_point = max(points, key=lambda p: p[1])

        font = ImageFont.truetype("src/task_4/data/font.ttf", 100)
        im = Image.new('RGB', (1000, 1000), ImageColor.getcolor("white",
"RGB"))\

        draw = ImageDraw.Draw(im)
```

```
        draw.polygon(
            xy=tuple(p for p in points),
            fill=self.__color.color, outline=(0, 0, 0)
        )
        w, h = draw.textsize(self.name, font=font)
        draw.text(
            xy=((1000 - w) / 2, lowest_point[1] + 50),
            text=self.name,
            fill=ImageColor.getcolor("black", "RGB"),
            font=font,
        )

        with open("src/task_4/data/hexagon.png", "wb") as file:
            im.save(file, format="png")

        im.show()
```

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).

а) Библиотека NumPy.

1. Создание массива. Функции array() и values().
2. Функции создания массива заданного вида.
3. Индексирование массивов NumPy. Индекс и срез.
4. Операции с массивами. Универсальные (поэлементные) функции.

б) Математические и статистические операции.

1. Функция mean()
2. Функция median()
3. Функция corrcoef()
4. Дисперсия var().
5. Стандартное отклонение std()

27. Вставьте первую строку после строки, в которой находится первый встреченный минимальный элемент. Вычислить значение медианы первой строки. Вычисление медианы выполнить двумя способами: через стандартную функцию и через программирование формулы.

```
import numpy
import numpy as np


class MatrixOperator:
    """Class for operating with random matrix."""

    def __init__(self):
```

```
        self.__matrix = np.ndarray([0, 0])

    def generateMatrix(self, a: int, b: int):
        self.__matrix = np.random.rand(a, b)

    def __str__(self):
        for row in self.__matrix:
            print(row)

    def insert_raw_after_the_raw_with_min_element(self):
        minimal = 0
        for index in range(0, self.__matrix.shape[0]):
            if self.__matrix[index, 0] < self.__matrix[minimal, 0]:
                self.__matrix = numpy.insert(self.__matrix, index + 1,
self.__matrix[0], axis=0)
                break

    @property
    def matrix(self):
        return self.__matrix

    @matrix.setter
    def matrix(self, new_matrix):
        self.__matrix = new_matrix

    def calculateMedianNp(self):
        return numpy.median(self.__matrix)

    def calculateMedian(self):
        r, c = self.__matrix.shape
        amount = r * c
        vals = numpy.sort(self.__matrix.reshape(amount))
        if amount % 2 == 0:
            return numpy.take(vals, [amount // 2, amount // 2 - 1]).mean()
        else:
            return vals[amount // 2]
```