Лабораторная работа №4.
Гимпель Кирилл гр. 253505
Вариант №5

Меню с выбором запускаемой задачи:

```python
# Author: Gimpel Kirill
# Date: 06.05.2024
# Version: 1.0

from task1.task1 import task1
from task2.task2 import task2
from task3.task3 import task3
from task4.task4 import task4
from task5.task5 import task5
from utils.secure_input import secure_input


if __name__ == '__main__':
    while True:
        choice = secure_input(int, "Enter task number or press 0 to exit: ", lambda x: 0 <= x <= 5)
        match choice:
            case 0:
                break
            case 1:
                task1()
            case 2:
                task2()
            case 3:
                task3()
            case 4:
                task4()
            case 5:
                task5()
```

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

5.	Реализуйте школьную нагрузку (фамилия преподавателя, класс, часы). Составьте программу, определяющую нагрузку каждого преподавателя. Определить, у какого преподавателя самая большая нагрузка и у какого самая низкая. Выведите нагрузку преподавателя, введенного с клавиатуры

```python
class Workload:
    def __init__(self, surname, name_class, amount_hours):
        self.surname = surname
        self.name_class = name_class
        self.amount_hours = amount_hours

    def __str__(self):
        return f"Workload({self.surname},{self.name_class},{self.amount_hours})"

    def __repr__(self):
        return self.__str__()
```

```python
from abc import ABC, abstractmethod
from .workload import Workload


class Serializer(ABC):
    @classmethod
    @abstractmethod
    def serialize(cls, workloads: dict[str, Workload], path: str):
        pass

    @classmethod
    @abstractmethod
    def deserialize(cls, path: str):
        pass
```

```python
from .workload import Workload


def find_workload_by_surname(all_workloads: dict[str, list[Workload]], surname: str):
    if surname in all_workloads:
        return sum(workload.amount_hours for workload in all_workloads[surname])
    return 0
```

```python
from .serializer import Serializer
from .workload import Workload
import csv


class CsvSerializer(Serializer):
    @classmethod
    def serialize(cls, all_workloads: dict[str, list[Workload]], path: str):
        try:
            with open(path, mode='w', newline='') as file:
                writer = csv.writer(file)
                writer.writerow(['Surname', 'Name class', 'Amount hours'])
                for surname, teacher_workloads in all_workloads.items():
                    for workload in teacher_workloads:
                        writer.writerow([surname, workload.name_class, workload.amount_hours])
        except:
            print("There was a problem with the serialisation of the object.")
```

```python
    @classmethod
    def deserialize(cls, path: str):
        all_worloads = {}
        try:
            with open(path, mode='r', newline='') as file:
                reader = csv.reader(file)
                next(reader)
                for row in reader:
                    workload = Workload(row[0], row[1], int(row[2]))
                    if workload.surname in all_worloads:
                        all_worloads[workload.surname].append(workload)
                    else:
                        all_worloads[workload.surname] = [workload]
        except:
            print("There was a problem with the deserialisation of the object.")
        return all_worloads
```

```python
from .serializer import Serializer
from .workload import Workload
import pickle


class PickleSerializer(Serializer):
    @classmethod
    def serialize(cls, all_workloads: dict[str, list[Workload]], path: str):
        try:
            with open(path, mode='wb') as file:
                pickle.dump(all_workloads, file)
        except:
            print("There was a problem with the serialisation of the object.")


    @classmethod
    def deserialize(cls, path: str):
        all_worloads = {}
        try:
            with open(path, mode='rb') as file:
                all_worloads = pickle.load(file)
        except:
            print("There was a problem with the deserialisation of the object.")
        return all_worloads
```

```python
def task1():
    all_workloads = {
        "Fedu": [Workload("Fedu", "11", 50), Workload("Fedu", "7", 40)],
        "Kirill": [Workload("Kirill", "9", 60)],
        "Sveta": [Workload("Sveta", "10A", 100)],
    }
    choice = secure_input(int, "Enter the serialisation method (1 - csv, 2 - pickle): ", lambda x: 1 <= x <= 2)
    serializer: Serializer
    match choice:
        case 1:
            path = "task1/task1.csv"
            serializer = CsvSerializer
        case 2:
            path = "task1/task1.pickle"
            serializer = PickleSerializer

    serializer.serialize(all_workloads, path)
    loaded_workloads = serializer.deserialize(path)
    print("Full school workload")
    pprint(loaded_workloads)
    surname = input("Enter the instructor's last name to find out their workload: ")
    print("Teacher workload:", find_workload_by_surname(loaded_workloads, surname))
```

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 (считаются только слова);
- среднюю длину слова в тексте в символах;

5.	Вывести все слова, начинающиеся со строчной буквы и знаки препинания. Определить, является ли заданная строка правильным MAC-адресом. Пример правильного выражения: aE:dC:cA:56:76:54. Пример неправильного выражения: 01:23:45:67:89:Az. определить количество слов в строке; найти самое длинное слово и его порядковый номер; вывести каждое нечетное слово

- 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 меньше одной) одинаковых скобок из следующего набора: «(», «)», «[», «]»;

  внутри смайлика не может встречаться никаких других символов. Например, эта последовательность является смайликом: «;---------[[[[[[[[». Эти последовательности смайликами не являются: «]», «;--»,«:»,«)».

```
class Analyser:
  def __init__(self, text: str = ""):
    self.__text = text

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


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


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

```
def interrogative_sentences_count(self):
  interrogative_sentences = re.findall("[A-Z0-9][^?!.]*?[?]+", self.__text)
  return len(interrogative_sentences)


def exclamation_sentences_count(self):
  exclamation_sentences = re.findall("[A-Z0-9][^?!.]*?[!]+", self.__text)
  return len(exclamation_sentences)


def narrative_sentences_count(self):
  narrative_sentences = re.findall("[A-Z0-9][^?!.]*?[.]+", self.__text)
  return len(narrative_sentences)


def word_count(self):
  return len(self.get_all_words())
```

```
def average_sentence_length(self):
  sentences = re.findall("[A-Z0-9][^?!.]*?[?.!]", self.__text)
  sum_lengths = sum(len(sentence) for sentence in sentences)
  return sum_lengths / len(sentences)


def average_word_length(self):
  words = self.get_all_words()
  sum_lengths = sum(len(word) for word in words)
  return sum_lengths / len(words)


def smileys_count(self):
  smileys = re.findall(r"(:|;)(\-*?)((\(+)|(\)+)|(\[+)|(\]+))", self.__text)
  return len(smileys)


def get_all_words(self):
  words = [word for word in re.findall(r"\b\w+\b", self.__text)]
  return words
```

```python
def get_all_words_start_lowercase(self):
    return list(filter(lambda word: word[0].islower(), self.get_all_words()))


def max_word_by_length(self):
    words = self.get_all_words()
    word = max(words, key=len)
    return word, words.index(word)


def get_odd_length_words(self):
    words = self.get_all_words()
    return list(filter(lambda word: len(word) % 2 == 1, words))
```

```python
def get_all_punctuation(self):
    text = re.sub(r"(:|;)(-*?)((\(+)|(\)+)|(\[+)|(\]+))", "", self.__text)
    puctuations = re.findall(r"([^\w\s])+", text)
    return {puctuation for puctuation in puctuations}


def get_all_mac_addresses(self):
    mac_addresses = re.findall(r"(?:[0-9a-fA-F]{2}[:-]){5}[0-9a-fA-F]{2}", self.__text)
    return [mac for mac in mac_addresses]
```

```python
def task2():
    try:
        analyser = Analyser()
        with open("task2/text.txt") as file:
            analyser.text = file.read()
        with open("task2/result.txt", 'w') as file:
            print("Sentences count:", analyser.sentences_count(), file=file)
            print("Interrogative sentences count:", analyser.interrogative_sentences_count(), file=file)
            print("Exclamation sentences count:", analyser.exclamation_sentences_count(), file=file)
            print("Narrative sentences count:", analyser.narrative_sentences_count(), file=file)
            print("Average sentence length:", analyser.average_sentence_length(), file=file)
            print("Average word length:", analyser.average_word_length(), file=file)
            print("Smileys count:", analyser.smileys_count(), file=file)
            print("Word count:", analyser.word_count(), file=file)
            max_word, index = analyser.max_word_by_length()
            print("Maximum word length:", max_word, file=file)
            print("Index maximum word length:", index, file=file)
            print("All the words that start with a lowercase letter:", analyser.get_all_words_start_lowercase(), file=file)
            print("All punctuation:", analyser.get_all_punctuation(), file=file)
            print("Odd length words:", analyser.get_odd_length_words(), file=file)
            print("All mac addresses:", analyser.get_all_mac_addresses(), file=file)

        with open("task2/result.txt") as file:
            print(file.read())

        with zipfile.ZipFile("task2/zipfile.zip", 'w') as zip:
            zip.write("task2/result.txt")
    except:
        print("Something went wrong. Try again later.")
```

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.

x	n	$F(x)$	$Math\ F(x)$	eps

Здесь x – значение аргумента, F(x) – значение функции, n – количество просуммированных членов ряда, Math F(x) – значение функции, вычисленное с помощью модуля math.

в) сохранить графики в файл

5.	$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots$

```python
def taylor_exp(x: float, n: int, create_list: bool = False):
    i = 0
    term = 1.0
    sum = term
    li = [term]
    while i + 1 < n:
        i += 1
        term *= x / i
        sum += term
        if create_list:
            li.append(term)

    if create_list:
        return sum, li
    return sum
```

```python
import matplotlib.pyplot as plt


def save_graph(path, x1, y1, x2, y2):
    plt.plot(x1, y1, label='math.exp', color="r")
    plt.plot(x2, y2, label='taylor_exp', color="g")
    plt.axis('equal')
    plt.legend()
    plt.savefig(path)
```

```python
def task3():
    x = secure_input(float, "Enter x: ")
    n = secure_input(int, "Enter n: ", lambda val: val > 1)
    f, arr = taylor_exp(x, n, True)
    math_f = math.exp(x)
    print("x | n | F(x) | Math F(x) | eps")
    print(f"{x} | {n} | {f} | {math_f} | {math.fabs(f - math_f)}")
    print(f"average: {statistics.mean(arr)}")
    print(f"median: {statistics.median(arr)}")
    print(f"mode: {statistics.mode(arr)}")
    print(f"variance: {statistics.variance(arr)}")
    print(f"standard deviation: {statistics.stdev(arr)}")

    x = numpy.arange(-1, 1, 0.05)
    y1 = [taylor_exp(xi, n) for xi in x]
    y2 = [math.exp(xi) for xi in x]
    save_graph("task3/graph.png", x, y1, x, y2)
```

Задание 4. В соответствии с заданием своего варианта разработать базовые классы и классы наследники.

Требования по использованию классов:

Абстрактный класс «Геометрическая фигура» содержит абстрактный метод для вычисления площади фигуры (https://docs.python.org/3/library/abc.html)

Класс «Цвет фигуры» содержит свойство для описания цвета геометрической фигуры (https://docs.python.org/3/library/functions.html#property)

Класс «Прямоугольник» (Круг, Ромб, Квадрат, Треугольник и т.д.) наследуется от класса «Геометрическая фигура». Класс должен содержать конструктор по параметрам «ширина», «высота» (для другого типа фигуры соответствующие параметры, например, для круга задаем «радиус») и «цвет». В конструкторе создается объект класса «Цвет фигуры» для хранения цвета. Класс должен переопределять метод, вычисляющий площадь фигуры https://docs.python.org/3/library/math.html .

Для класса «Прямоугольник»(тип фигуры в инд. задании)

определить метод, который возвращает в виде строки основные параметры фигуры, ее цвет и площадь. Использовать метод format (https://pyformat.info/)

название фигуры должно задаваться в виде поля данных класса и возвращаться методом класса.

В корневом каталоге проекта создайте файл main.py для тестирования классов. Используйте конструкцию, описанную в https://docs.python.org/3/library/__main__.html

Пример объекта: Прямоугольник синего цвета шириной 5 и высотой 8.

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.

5.	Построить треугольник по стороне a и двум прилежащим к ней углам B и C (в градусах).

```python
class Point:
  def __init__(self, x, y):
    self.__x = x
    self.__y = y

  @property
  def x(self):
    return self.__x


  @x.setter
  def x(self, x):
    self.__x = x


  @property
  def y(self):
    return self.__y


  @y.setter
  def y(self, y):
    self.__y = y


  def distance(self, other):
    return sqrt((self.__x - other.__x)**2 + (self.__y - other.__y)**2)
```

```python
class NameableFigure:
  def __init__(self, name: str):
    self.__name = name


  @property
  def name(self):
    return self.__name
```

```python
class FigureColor:
    def __init__(self, color: str):
        self.__color = color


    @property
    def color(self):
        return self.__color


    @color.setter
    def color(self, color: str):
        self.__color = color
```

```python
from abc import ABC, abstractmethod


class GeometricFigure(ABC):
    @abstractmethod
    def area():
        pass
```

```python
class Triangle(NameableFigure, GeometricFigure):
    def __init__(self, side, angleB, angleC, color, start = Point(1, 1)):
        super().__init__("Triangle")
        self.__color = FigureColor(color)
        self.__start = start
        self.__B = Point(start.x, start.y)
        self.__C = Point(start.x + side, start.y)
        AC = side * math.sin(math.radians(angleB)) / math.sin(math.radians(180 - angleB - angleC))
        self.__A = Point(self.__C.x - AC * math.cos(math.radians(angleC)),
                         self.__C.y + AC * math.sin(math.radians(angleC)))
```

```python
def save_and_show(self, path: str, description: str):
    x = [self.__A.x, self.__B.x, self.__C.x, self.__A.x]
    y = [self.__A.y, self.__B.y, self.__C.y, self.__A.y]
    plt.plot(x, y, label=description, color=self.__color.color)
    plt.legend()
    plt.fill(x, y, color=self.__color.color)
    plt.axis('equal')
    plt.ylim(self.__start.y - 1, self.__A.y + 1)
    plt.savefig(path)
    plt.show()


def area(self):
    return math.fabs(0.5*(self.__A.x*(self.__B.y - self.__C.y) +
                          self.__B.x*(self.__C.y - self.__A.y) +
                          self.__C.x*(self.__A.y - self.__B.y)))
```

```python
    def __str__(self):
        return '''
Figure: {}
Side AC = {:.5f}
Side AB = {:.5f}
Side BC = {:.5f}
Point A = ({:.5f}, {:.5f})
Point B = ({:.5f}, {:.5f})
Point C = ({:.5f}, {:.5f})
Area S = {:.5f}
        '''.format(
            self.name,
            self.__A.distance(self.__C),
            self.__A.distance(self.__B),
            self.__B.distance(self.__C),
            self.__A.x, self.__A.y,
            self.__B.x, self.__B.y,
            self.__C.x, self.__C.y,
            self.area()
        )
```

```python
def task4():
    side = secure_input(float, "Enter a side: ", lambda x: x > 0)
    angleB = secure_input(float, "Enter angle B: ", lambda x: 0 < x < 180)
    angleC = secure_input(float, "Enter angle C: ", lambda x: 0 < x < 180 - angleB)
    colors = {'r', 'b', 'c', 'g', 'm', 'y', 'k'}
    print("Acceptable colors:", colors)
    color = secure_input(str, "Enter a color: ", lambda x: x in colors)
    description = secure_input(str, "Enter description: ", lambda x: len(x) > 0)
    triangle = Triangle(side, angleB, angleC, color)
    print(str(triangle))
    triangle.save_and_show("task4/graph.png", description)
```

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()

5.	Отсортировать по возрастанию элементы последней строки матрицы. Вычислить значение медианы этой строки матрицы. Вычисление медианы выполнить двумя способами: через стандартную функцию и через программирование формулы.

```python
class MatrixWorker:
    def __init__(self, low: int, high: int):
        size = np.random.randint(1, 11, size=2)
        self.__matrix = np.random.randint(low, high, size=size)


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


    def sort_last_row(self):
        self.__matrix[-1] = np.sort(self.__matrix[-1])
```

```python
    def median_last_row(self):
        is_sorted = np.all(np.diff(self.__matrix[-1]) >= 0)

        if not is_sorted:
            row = np.sort(self.__matrix[-1])
        else:
            row = self.__matrix[-1]
        mid = (row.size - 1) // 2
        if row.size % 2 == 1:
            return row[mid]
        return (row[mid] + row[mid + 1]) / 2


    def median_last_row_numpy(self):
        return np.median(self.__matrix[-1])
```

```python
from .matrix_worker import MatrixWorker


def task5():
    worker = MatrixWorker(1, 100)
    print(worker.matrix)
    print("\nAfter sorting the last row")
    worker.sort_last_row()
    print(worker.matrix)
    print("\nMedian:", worker.median_last_row())
    print("Median (numpy):", worker.median_last_row_numpy())
```

Проверка на ввод:

```python
def secure_input(data_type: type, message: str, func: Callable[[type], bool] = None) -> type:
    """
    This function ensures the user inputs data of the correct type.
    If the user inputs data of the wrong type, it will keep prompting
    the user for input until they input data of the correct type.

    Args:
        data_type (type): The type of the data that the user is supposed to input.
        message (str): The message that is displayed to the user when prompting for input.
        func (function, optional): If this function returns False, the user will be prompted
        for input again.

    Returns:
        data_type: The user's input, converted to the specified data type.
    """
    again = True
    while again:
        try:
            data = data_type(input(message))
            again = not func(data) if func else False
        except ValueError:
            pass
        if again:
            print("Incorrect input!!!")
    return data
```