|  |  |
| --- | --- |
| **К Г Э У** | МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**  (ФГБОУ ВО «КГЭУ») |

**Кафедра «Информатики и информационных управляющих систем»**

ОТЧЕТ

Дисциплина «Алгоритмизация и программирование»

Выполнил: Соловьёв Л.А.  
Группа: ПИ-1-22  
Преподаватель: Шорина Т.В.

Казань, 2024

***Постановка задачи***

1. Изучение работы функций библиотеки SciPy

SciPy — это библиотека Python с открытым исходным кодом, предназначенная для решения научных и математических проблем. Она построена на базе NumPy и позволяет управлять данными, а также визуализировать их с помощью разных высокоуровневых команд.

**Основные команды модуля SciPy**

|  |  |
| --- | --- |
| Функция | Описание |
| cluster | Алгоритмы кластерного анализа |
| constants | Физические и математические константы |
| fftpack | Быстрое преобразование Фурье |
| integrate | Решения интегральных и обычных дифференциальных уравнений |
| interpolate | Интерполяция и сглаживание сплайнов |
| io | Ввод и вывод |
| linalg | Линейная алгебра |
| odr | Метод ортогональных расстояний |
| optimize | Оптимизация и численное решение уравнений |
| signal | Обработка сигналов |

**Примеры обработки данных с использованием модуля**

**Задача 1.** Интерполяция данных по температурным значениям и времени

Алгоритм выполнения:

Для начала, импортируем функцию interp1d из библиотеки scipy.interpolate с помощью команды from scipy.interpolate import inter1d, а так же библиотеку numpy. Затем создаем переменные times и temperatures. После этого, используя функцию interp1d() интерполируем функцию температуры от времени. Затем, выводим полученный результат с помощью функции print.

Ожидаемый итог:

После выполнения кода ожидается, что на выходе мы получим график функции y = x2.

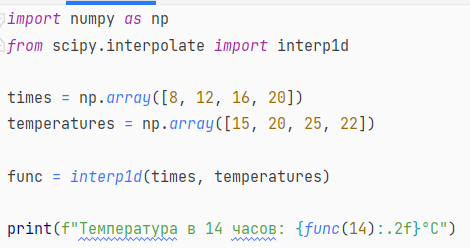


Рисунок 1.1 – Код к задаче №1



Рисунок 1.2 – Вывод к задаче №1

**Задача 2.** Решение системы уравнений

Алгоритм выполнения:

Для начала, импортируем модуль scipy.linalg с помощью import scipy.linalg, а так же библиотеку numpy. Затем создаем переменные A и b, означающие список со списками всех коэффициентов перед переменными каждого уравнения и значения уравнения. После этого, используя функцию scipy.linalg.solve() вычисляем решение уравнения. Затем, выводим полученный результат с помощью функции print.

Ожидаемый итог:

После выполнения кода ожидается, что на выходе мы получим решение системы уравнений.

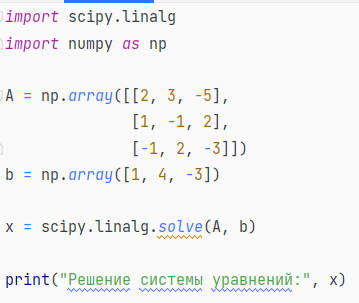


Рисунок 2.1 – Код к задаче №2



Рисунок 2.2 – Вывод к задаче №2

**Задача 3.** Решение кубического уравнения

Алгоритм выполнения:

Для начала, импортируем модуль scipy.optimize с помощью import scipy.optimize. Затем создаем переменную roots и, используя функцию scipy.optimize.fsolve() вычисляем решение уравнения. Затем, выводим полученный результат с помощью функции print.

Ожидаемый итог:

После выполнения кода ожидается, что на выходе мы получим решения кубического уравнения.

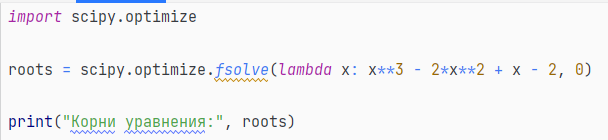


Рисунок 3.1 – Код к задаче №3



Рисунок 3.2 – Вывод к задаче №3

**Задача 4.** Вычисление определенного интеграла

Алгоритм выполнения:

Для начала, импортируем модуль scipy.optimize с помощью import scipy.optimize. Затем создаем переменную roots и, используя функцию scipy.optimize.fsolve() вычисляем решение уравнения. Затем, выводим полученный результат с помощью функции print.

Ожидаемый итог:

После выполнения кода ожидается, что на выходе мы получим значение определенного интеграла.

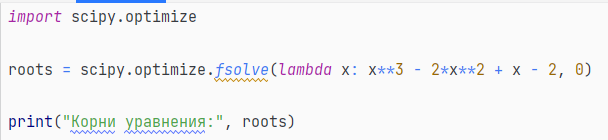


Рисунок 4.1 – Код к задаче №4



Рисунок 4.2 – Вывод к задаче №4

**Задача 5.** Получение физических констант

Алгоритм выполнения:

Для начала, импортируем из модуля scipy.constants значения постоянных скорости света, гравитационной и Планка с помощью from scipy.constants import c, G, h. Затем, выводим полученные константы с помощью функции print.

Ожидаемый итог:

После выполнения кода ожидается, что на выходе мы получим решения кубического уравнения.

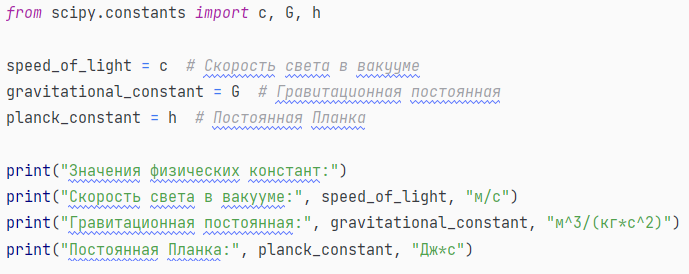


Рисунок 5.1 – Код к задаче №5

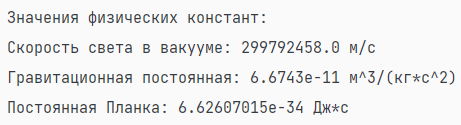


Рисунок 5.2 – Вывод к задаче №5

**Задача 6.** Получение собственных значений и векторов матрицы

Алгоритм выполнения:

Для начала, импортируем модуль scipy.linalg с помощью import scipy.linalg. После зададим переменной A значение матрицы. Затем создаем переменные eigenvalues и eigenvectors, используя функцию scipy.linalg.eig() вычисляем собственные вектора и значения. Затем, выводим полученный результат с помощью цикла функции print.

Ожидаемый итог:

После выполнения кода ожидается, что на выходе мы получим собственные значения и векторы матрица.



Рисунок 6.1 – Код к задаче №6

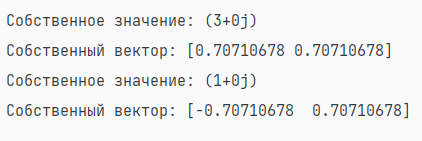


Рисунок 6.2 – Вывод к задаче №6

**Вывод**

SciPy - незаменимый инструмент для научных и технических вычислений в Python. Она предоставляет широкий спектр функций, которые позволяют пользователям решать сложные проблемы в различных областях, экономя время и усилия.

**Список использованных источников**

1. <https://matplotlib.org/cheatsheets/>