МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

**ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ PYTHON. РЕАЛИЗАЦИЯ**

**ШИФР С ОМОФОНАМИ**

Пояснительная записка

Отчёт по дисциплине

«Учебная практика»

ТПЖА 090302.293ПЗ

Разработал студент гр.ИТб-1301-01-00 \_\_\_\_\_\_\_\_\_\_\_\_/Бессонов И.А./

(подпись)

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_/ Родионов К.В./

(подпись)

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_»\_\_\_\_\_\_\_\_\_\_2021 г.

Киров 2021

**Задания:**

1. Выполнить обучающие задания, приведённые в методических указаниях, для ознакомления с методами работы с языком Python.

2. Реализовать криптографический алгоритм (Шифр с омофонами) при помощи языка Python, не используя библиотечные функции связанные непосредственно с шифрованием. Разработанная программа должна осуществлять шифрование и дешифрование текста в соответствии с вариантом. В приложении должна быть возможность ввода текста (вручную или загрузка из файла – на усмотрение разработчика), ключа (если этого требует алгоритм) и зашифрованного текста.

Реферат

Бессонов И.А. Программирование на языке Python: ТПЖА. 090302.293 ПЗ: Отчет по учебной практике / ВятГУ, каф. САУ; рук. К.В. Родионов. – Киров, 2021. ПЗ 33 с., 14 рис., 1 табл., 3 источников,5прил.

PYTHON, ПРОГРАММИРОВАНИЕ, ФУНКЦИИ, ПЕРЕМЕННЫЕ, ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ.

Объектом исследования является высокоуровневый язык программирования Python, разработка с использованием современных информационных технологий.

Цель работы – изучение языка программирования Python.

Реализованы приложения в среде программирования Spyder, предусмотрена визуализация.

Разработано приложение выполняющее шифрование и дешифрование данных по криптографическому алгоритму Шифр с омофонами, проанализирована задача и описаны результаты полученной программы.

|  |  |
| --- | --- |
| **Содержание**  Изм.  Лист  № докум.  Подпись  Дата  Лист  1  ТПЖА.270304.010 ПЗ  Разраб.  *Шевелёв И.А.*  Провер.  Родионов К.В.  Реценз.  Н. Контр.  Утверд.  Программирование на языке Python. Реализация <ВЕЛИКИЙ ШИФР>  Литер  Листов  33  *Кафедра САУ,*  *гр. УТб-1301-02-00* |  |
| Введение……………………………………………………………………….. | 2 |
| Обзор среды программирования Spyder и введение в Python……………… | 3 |
| 1.1.Среда программирования Spyder………………………………………… | 3 |
| 1.2.Обзор языка программирования Python…………………………………. | 4 |
| Изучение синтаксиса Python, с помощью выполнения заданий…………… | 6 |
| 2.1. Задание 1. Алгоритм кластеризации k-means 1………………………… | 6 |
| 2.2. Задание 2. Классификация k-ближайших соседей…………………….. | 7 |
| 2.3. Задание 3. Работа с математической статистикой……………………... | 9 |
| Криптографический алгоритм «Великий Шифр»…………………...……… | 15 |
| 3.1 История алгоритма «Великий Шифр»……………………………….. | 15 |
| 3.2 Шифрование и дешифрование исходных данных……….………...… | 15 |
| Заключение………………………………………………………………..…… | 17 |
| Приложение А (обязательное) Листинг программного кода реализации алгоритма k-means…………………………………….………………………. | 18 |
| Приложение Б (обязательное) Листинг программного кода реализации алгоритма классификации k-NN………………………………………...…… | 20 |
| Приложение В (обязательное) Листинг программного кода реализации задания с массивами и графиками…………………………………………… | 22 |
| Приложение Г (обязательное) Листинг программного кода реализации алгоритма шифрования «Великий Шифр»………………………………..… | 25 |
| Приложение Д (справочное) Библиографический список………………..... | 33 |

# Введение

Учебная практика является логическим продолжением летней экзаменационной сессии, позволяя студентам получить дополнительные практические навыки на предприятиях или новые теоретические знания, которые могут помочь в дальнейшем учебном процессе или даже выборе работы.

Знания, которые получают студенты во время основного курса обучения, не всегда достаточны при выборе рода деятельности по окончании университета, именно поэтому летняя практика – хороший способ получить дополнительные навыки, выходящие за рамки курса.

Целью данной практики является изучение основ работы с языком программирования Python и выполнение заданий.

Задачами практики являлись:

1. Изучение среды программирования Spyder.
2. Изучение языка программирования Python.
3. Изучение алгоритма работы криптографического шифра «Великий Шифр».
4. Заключение о рациональном использовании данного метода шифрования.

# **Обзор среды программирования Spyder и введение в Python.**

Для комфортного использования языка Pythonнужно изучить среду программирования для этого языка.

* 1. Среда программирования Spyder

Spyder – мощная научная среда, написанная на Python для Python и разработанная учеными, инженерами и аналитиками данных и для них. Он предлагает уникальное сочетание передовых функций редактирования, анализа, отладки и профилирования комплексного инструмента разработки с исследованием данных, интерактивным выполнением, глубоким осмотром и прекрасными возможностями визуализации научного пакета.Python может поддерживать широкий перечень стилей разработки приложений, в том числе, очень удобен для работы с ООП и функциональным программированием.

Интерфейс среды разработки Spyderпредставлен на рисунке 1.1.1.

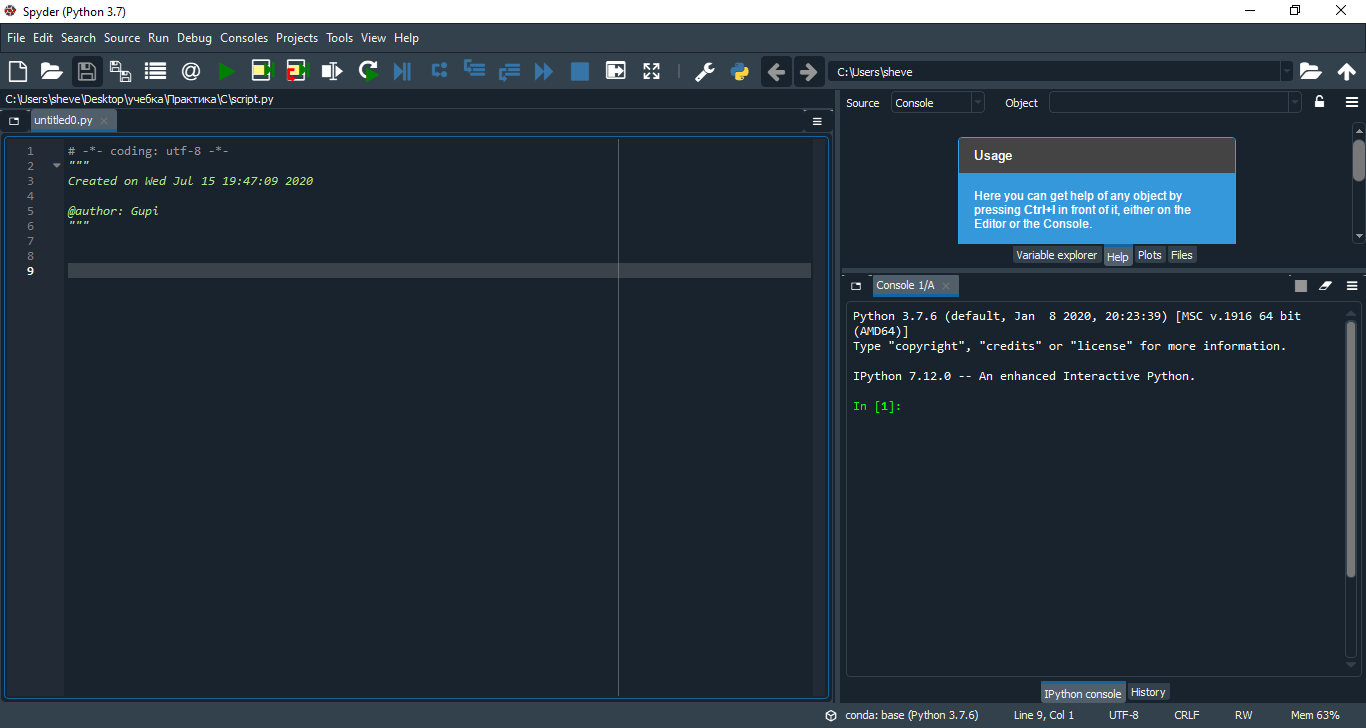


Рисунок 1.1.1 – Интерфейс Spyder

Помимо множества встроенных функций, его возможности могут быть расширены с помощью системы плагинов и API. Кроме того, Spyder также можно использовать в качестве библиотеки расширений PyQt5, что позволяет разработчикам использовать его функциональные возможности и встраивать его компоненты, такие как интерактивная консоль, в свое собственное программное обеспечение PyQt [1, c. 4].

* 1. **Обзор языка программирования Python**

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Python поддерживает [структурное](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [обобщенное](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [объектно-ориентированное](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [функциональное](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [аспектно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Основные архитектурные черты–[динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F),[автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), полная[интроспекция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), механизм[обработки исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка[многопоточных вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), высокоуровневые[структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на[модули](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые, в свою очередь, могут объединяться в пакеты.

Эталонной реализацией Python является интерпретатор[CPython](https://ru.wikipedia.org/wiki/CPython), поддерживающий большинство активно используемых платформ. Он распространяется под[свободной лицензией](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая[проприетарные](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E). Есть[реализация интерпретатора для JVM](https://ru.wikipedia.org/wiki/Jython)с возможностью[компиляции](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80),[CLR](https://ru.wikipedia.org/wiki/IronPython), [LLVM](https://ru.wikipedia.org/wiki/LLVM), другие независимые реализации. Проект[PyPy](https://ru.wikipedia.org/wiki/PyPy)использует[JIT-компиляцию](https://ru.wikipedia.org/wiki/JIT-%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F), которая значительно увеличивает скорость выполнения Python-программ.

Python –активно развивающийся[язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), новые версии с добавлением/изменением языковых свойств выходят примерно раз в два с половиной года. Язык не подвергался официальной стандартизации, роль стандарта де-факто выполняет[CPython](https://ru.wikipedia.org/wiki/CPython), разрабатываемый под контролем автора языка. В настоящий момент Python занимает третье место в[рейтинге TIOBE](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_TIOBE) с показателем 10,2%. Аналитики отмечают, что это самый высокий балл Python за все время его присутствия в рейтинге[2, c. 5].

# Изучение синтаксиса Python, с помощью выполнения заданий

Следующей частью практической работы было получение минимальных знания языка Python. Для получения минимальных знаний были предоставлены задания .

* 1. **Задание 1. Алгоритм кластеризации k-means**

В задании 1 требуется реализовать алгоритм k-means, с помощью которого выделить два кластера, описывающих формируемые группы студентов.

Алгоритм работы k-meansпредставлен ниже:

1. Задается количество кластеров k, которые требуется обнаружить.

2. Центры кластеров изначально инициализируются случайным образом.

3. Каждый из объектов приписывается к ближайшему кластеру.

4. На основании объектов, вошедших в каждый кластер, центры кластеров пересчитываются.

5. Шаги 3 и 4 повторяются до тех пор, пока центры кластеров не стабилизируются, то есть на очередной итерации объекты будут принадлежать тем же кластерам, что и до этого.

График, полученный в результате работы программы, предоставлен на рисунке2.1.1.

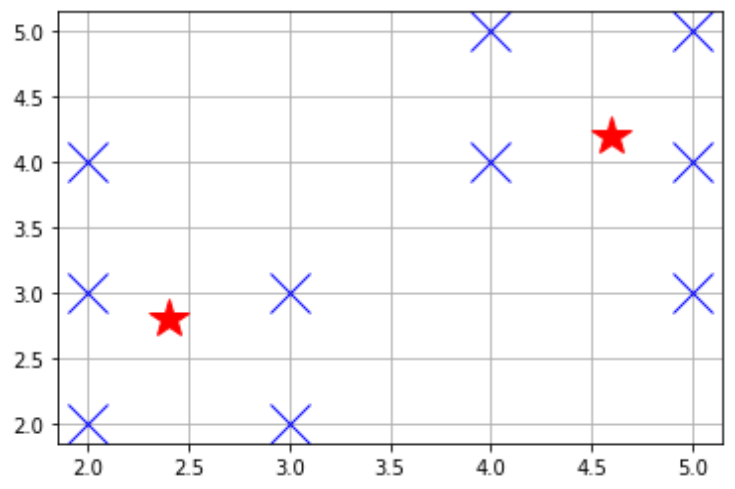


Рисунок 2.1.1 – Результат работы алгоритма k-means,

на примере задания 1

Полученный график показывает, что программа работает правильно.

Листинг программы представлен в приложение А.

* 1. **Задание 2. Классификация k-ближайших соседей**

В задании 2 требуется реализовать алгоритм k-NN, с помощью которого определить, к какому виду относятся обезьяны, изученные его ассистентом во время экспедиции.

Алгоритм работы k-NN представлен ниже:

* 1. Измеряем расстояние от объектаobj до каждого объекта в матрице X.
  2. Сортируем полученные значения в порядке увеличения расстояния (в начале будут объекты наиболее близкие к obj, в конце – наиболее удаленные).
  3. Берем первые k объектов из отсортированного списка, определяем, какой класс встречается там чаще всего. Именно этому классу и принадлежит наш объект obj.

Для реализации данного алгоритма была дана таблица с параметрами и видами некоторых обезьян.(Таблица 2.2.1).

Таблица 2.2.1 –Параметры и виды некоторых обезьян

|  |  |  |
| --- | --- | --- |
| **Рост** | **Вес** | **Вид** |
| 33 | 21 | Лемур |
| 41 | 13 | Лемур |
| 18 | 22 | Лемур |
| 38 | 34 | Лемур |
| 62 | 118 | Шимпанзе |
| 59 | 137 | Шимпанзе |
| 95 | 131 | Шимпанзе |
| 83 | 110 | Шимпанзе |
| 185 | 155 | Горилла |
| 193 | 129 | Горилла |
| 164 | 135 | Горилла |
| 205 | 131 | Горилла |
| 145 | 55 | Орангутанг |
| 168 | 35 | Орангутанг |
| 135 | 47 | Орангутанг |
| 138 | 66 | Орангутанг |

Результат работы программы представлен на рисунках 2.2.2.

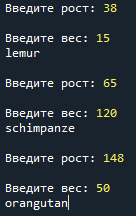


Рисунок 2.2.2 – Результат работы алгоритма k-NN,

на примере задания 2

По полученным результатам можно сделать вывод, что программа работает правильно.

Листинг программы представлен в приложение Б.

* 1. **Задание 3. Работас математической статистикой**

В задании 3 нужно было несколько подпрограмм.

Список заданий представлен ниже:

* 1. Создать в Python переменную, массив, матрицу с заданными, случайными целочисленными, нулевыми, единичными значениями.

Результат выполненной подпрограммы представлен на рисунке 2.3.1.

* 1. Импортировать переменную (матрицу) из текстовогоструктурированного файла (файл создать вручную: разделители столбцов – пробелы).

Результат выполненной подпрограммы представлен на рисунке 2.3.2.

* 1. Загрузить одномерные данные из каталога data/1D из файла с расширением .mat согласно варианту. Оценить параметры одномерной случайной величины, используя команды NumPy: np.max, np.min, np.median, np.mean, np.var, np.std; команды сохранить в скрипт.

Результат выполненной подпрограммы представлен на рисунке 2.3.3.

* 1. Вывести графики одномерных случайных величин и их плотности распределения. Команды сохранить в скрипт. Подобрать вид распределения одномерной случайной величины и его параметры. На одном графике отобразить случайную величину, уровень среднего значения и дисперсию.

Результат выполненной подпрограммы представлен на рисунке 2.3.4.1 – 2.3.4.3.

* 1. Построить и вывести на графике автокорреляцию заданной одномерной величины. Команды сохранить в скрипт.

Результат выполненной подпрограммы представлен на рисунке 2.3.5.

* 1. Импортировать из каталога data/NDфайл \*.matс многомерными даннымисогласно варианту. Первые 5 столбцов считать входными значениями (измерения датчиков, наблюдения и т.д.), последний 6 столбец – выходная величина, для которой требуется установить зависимость от входных величин.

Результат выполненной подпрограммы представлен на рисунке 2.3.6.

* 1. Построить матрицу корреляции для всех входных и выходных величин. Сделать выводы о зависимости/независимости выходной величины от каждой из входных компонент. Отобразить точечный график для случайных величин, коэффициент корреляции для которых по модулю больше 0.8.

Результат выполненной подпрограммы представлен на рисунке 2.3.7.1 – 2.3.7.2.

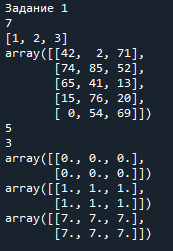


Рисунок 2.3.1 – Результат выполнения первой подпрограммы

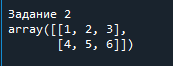


Рисунок 2.3.2 – Результат выполнения второй подпрограммы

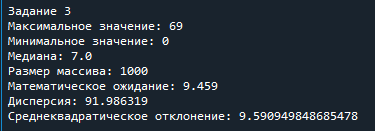


Рисунок 2.3.3 – Результат выполнения третьей подпрограммы

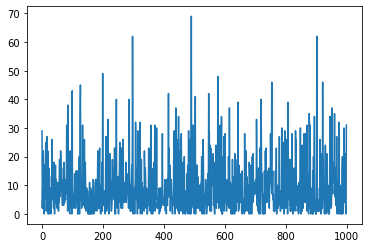


Рисунок 2.3.4.1 – Результат выполнения третьей подпрограммы. График одномерных случайных величин и их плотности распределения

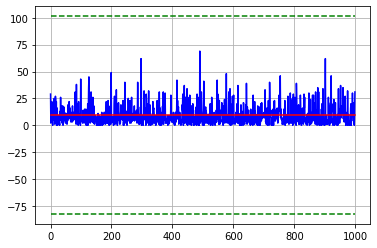


Рисунок 2.3.4.2 – Результат выполнения четвертой подпрограммы. График значений массива, среднего значения и дисперсии

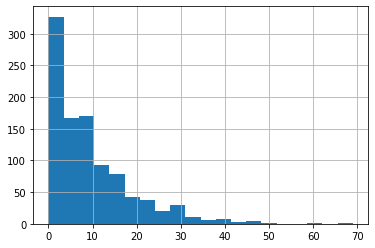


Рисунок 2.3.4.3 – Результат выполнения четвертой подпрограммы. График закона распределения случайной величины

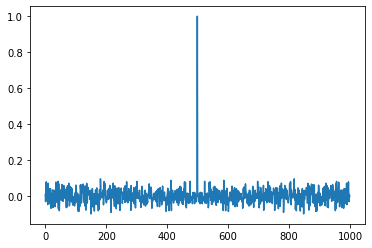


Рисунок 2.3.5– Результат выполнения пятой подпрограммы. График автокорреляции.

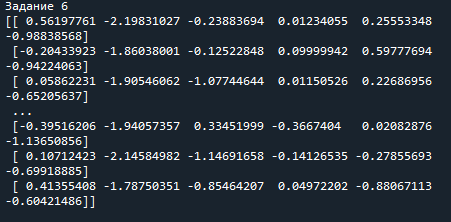


Рисунок 2.3.6– Результат выполнения шестой подпрограммы

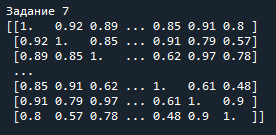


Рисунок 2.3.7.1– Результат выполнения седьмой подпрограммы. Матрица корреляции

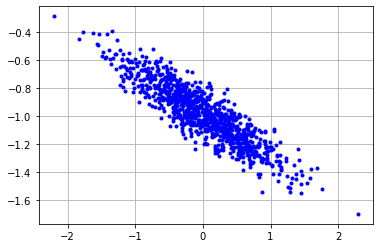


Рисунок 2.3.7.1– Результат выполнения седьмой подпрограммы.График значений столбцов с высоким коэффициентом корреляции

Из полученных результатов и графиков, можно сделать ввод, что программа работает правильно.

Листинг программы представлен в приложении В.

# 3 Криптографический алгоритм «Великий Шифр»

Перед написание программы шифрования и дешифрования текста с помощью криптографического алгоритма «Великий Шифр», нужно рассмотреть историю алгоритма и принцип шифрования и дешифрования.

**3.1 История алгоритма «Великий Шифр»**

Великий Шифр – шифр, разработанный [Антуаном Россиньолем](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%BD%D1%8C%D0%BE%D0%BB%D1%8C,_%D0%90%D0%BD%D1%82%D1%83%D0%B0%D0%BD) и его сыном Бонавентуром Россиньолем. Великий Шифр получил такое название из-за своего качества и репутации не взламываемого. Модифицированные формы использовались французской армией до лета 1811 года, а после того, как шифр перестали применять, многие документы во французских архивах стали нечитаемыми.

Великий шифр представляет собой набор из 587 чисел, большая часть которых обозначает слоги. Существуют различные варианты, в которых числами заменены также буквы или же целые слова. Применяемые для борьбы с частотным анализом некоторые числа являются ловушками, например, одно из чисел не заменяет ни слог, ни букву, а используется только для удаления предыдущего числа. Кроме того, во многие варианты шифра были включены альтернативные числа для наиболее используемых слогов и букв[3, c. 15].

**3.2 Шифрование и дешифрование исходных данных**

Алгоритм шифрования представлен ниже:

1. Исходные данные проверяются на наличие заранее записанных слогов и слов.
2. Если такие данные присутствуют, то их заменяют на заранее подготовленные символы, если нет таких данных, то пункт 2 пропускается.
3. Если после пункта 2 остались данные, которые не зашифрованы, они шифруются побуквенно с помощью заранее указанных цифр, у каждой буквы есть несколько заранее записанных цифр.
4. На все оставшиеся не занятые элементы памяти добавляются цифры ловушки (цифры к которым не присвоено никакое значение).
5. Получившиеся данные выводят.

При процессе дешифрования происходит замена цифр на приписанные им данные, место цифр-ловушек ничего не ставится.

Результат шифрования представлен на рисунке 3.3.2.

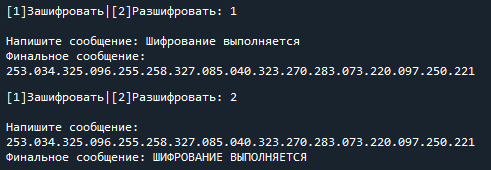


Рисунок 3.3.2 – Результаты шифрования и дешифрования предложения.

Листинг криптографического алгоритма «Великий Шифр»представлен в приложении Г.

# Заключение

После выполнения практики были получены знания в работе со средой программирования Spyder, а вместе с этим и начальные знания в языке программирования Python. После начального изучения языка Pythonстало понятно, что у него низкий порог вхождения и понятный синтаксис. Благодаря полученным теоретическим знаниям, можно легко продолжить изучение языка Python.

Так же были получены знания по криптографическим алгоритмам, благодаря которым стало понятнее работа шифровальных служб, что может помочь в шифровании своих личных данных.

**Приложение А**

(обязательное)

**Листинг программного кода реализации алгоритма k-means**

# #run.py

# import numpy as np

# import matplotlib.pyplot as plt

# from k\_means import kmeans

# X = np.array([

# [4, 4],

# [3, 3],

# [5, 3],

# [2, 3],

# [5, 5],

# [3, 2],

# [2, 4],

# [4, 5],

# [5, 4],

# [2, 2]])

# ans = kmeans(2, X)

# print(ans)

# plt.plot(X[:,0], X[:,1], 'bx', ans[:,0], ans[:,1], 'r\*', markersize=20)

# plt.grid()

# plt.show()

# # k\_means.py

# import numpy as np

# def dist(A, B):

# r = np.sqrt(np.sum((A - B)\*\*2))

# return r

# def class\_of\_each\_point(x, centers):

# m = len(x)

# k = len(centers)

# distances = np.zeros((m, k))

# for i in range(m):

# for j in range(k):

# distances[i, j] = dist(centers[j], x[i])

# return np.argmin(distances, axis=1)

# def kmeans(k, X):

# m = X.shape[0]

# n = X.shape[1]

# curr\_iteration = prev\_iteration = np.zeros(m)

# centers=(np.max(X, axis=0)-np.min(X, axis=0)\*np.random.random((k, n))-np.min(X, axis=0))

# curr\_iteration = class\_of\_each\_point(X, centers)

# while not np.all(curr\_iteration == prev\_iteration):

# prev\_iteration = curr\_iteration

# for i in range(k):

# sub\_X = X[curr\_iteration == i,:]

# if len(sub\_X) > 0:

# centers[i,:] = np.mean(sub\_X, axis=0)

# curr\_iteration = class\_of\_each\_point(X, centers)

# returncenters

**ПриложениеБ**

(обязательное)

**Листинг программного кода реализации алгоритма классификации**

**k-NN**

# #Run.py

# import numpy as np

# from kNN import k\_nearest

# X = np.array([

# [33, 21, 1],

# [41, 13, 1],

# [18, 22, 1],

# [38, 34, 1],

# [62, 118, 2],

# [59, 137, 2],

# [95, 131, 2],

# [83, 110, 2],

# [185, 155, 3],

# [193, 129, 3],

# [164, 135, 3],

# [205, 131, 3],

# [145, 55, 4],

# [168, 35, 4],

# [135, 47, 4],

# [138, 66, 4]])

# height = int(input('Введите рост: '))

# weight = int(input('Введите вес: '))

# obj = np.array([height, weight])

# # классификация методом k ближайших соседей

# k = 3

# object\_class = k\_nearest(X, k, obj)

# # вывод результата классификации

# monkeys = {1: 'lemur', 2: 'schimpanze', 3: 'gorilla', 4: 'orangutan'}

# print(monkeys[object\_class])

# #kNN.py

# import numpy as np

# import math

# def returnIndexOfMin(array):

# indexOfMin = 0

# a = array[0]

# for i in range(len(array)):

# if array[i] < a:

# a = array[i]

# indexOfMin = i

# return indexOfMin

# def k\_nearest(X, k, obj):

# sub\_X=X[:,[0,1]]

# distances=np.zeros(sub\_X.shape[0])

# for i in range(sub\_X.shape[0]):

# distances[i]= float(dist(sub\_X[i],obj))

# randomArray=np.zeros(k)

# for i in range(k):

# j=returnIndexOfMin(distances)

# randomArray[i] = X[j,2]

# distances[j]= math.inf

# nearest\_classes=np.ones(k)

# for i in range(k):

# nearest\_classes[i]=randomArray[i]

# unique, counts = np.unique(nearest\_classes, return\_counts=True)

# object\_class = unique[np.argmax(counts)]

# return object\_class

# def dist(p1, p2):

# return math.sqrt(sum((p1 - p2)\*\*2))

**ПриложениеB**

(обязательное)

**Листинг программного кода реализации задания с массивами и графиками**

# #Day3. py

# import pprint

# import numpy as np

# import scipy.io

# import matplotlib.pyplot as plt

# print('\nЗадание 1')

# a = 7

# b = [1, 2, 3]

# c = np.random.randint(100, size=(5,3))

# pprint.pprint(a)

# pprint.pprint(b)

# pprint.pprint(c)

# print(c.shape[0])

# print(c.shape[1])

# arr = np.ones((2, 3))

# arr7 = np.ones((2, 3)) \* 7

# zero = np.zeros((2, 3))

# pprint.pprint(zero)

# pprint.pprint(arr)

# pprint.pprint(arr7)

# print('\nЗадание 2')

# data=np.loadtxt('C:/Users/sheve/Desktop/test.txt', dtype=np.int32)

# pprint.pprint(data)

# print('\nЗадание 3')

# data = scipy.io.loadmat('C:/Users/sheve/Desktop/Учебка/Практика/D/data/1D/var2.mat')

# data = data['p']

# data = data.astype(int)

# print ("Максимальное значение:", np.max(data))

# print ("Минимальное значение:", np.min(data))

# print ("Медиана:", np.median(data))

# print ("Размер массива:", len(data))

# print ("Математическое ожидание:", np.mean(data))

# print ("Дисперсия:", np.var(data))

# print ("Среднеквадратическое отклонение:", np.std(data))

# print('\nЗадание 4')

# plt.plot(data)

# plt.show()

# mean = np.mean(data) \* np.ones(len(data))

# var = np.var(data) \* np.ones(len(data))

# plt.plot(data, 'b-', mean, 'r-', mean-var, 'g--', mean+var, 'g--')

# plt.grid()

# plt.show()

# plt.hist(data, bins=20)

# plt.grid()

# plt.show()

# print('\nЗадание 5')

# def autocorrelate(a):

# n = len(a)

# cor = []

# for i in range(n//2, n//2+n):

# a1 = a[:i+1] if i< n else a[i-n+1:]

# a2 = a[n-i-1:] if i< n else a[:2\*n-i-1]

# cor.append(np.corrcoef(a1, a2)[0, 1])

# return cor

# print(data.shape)

# data = np.ravel(data)

# print(data.shape)

# cor = autocorrelate(data)

# plt.plot(cor)

# plt.show()

# print('\nЗадание 6')

# data = scipy.io.loadmat('C:/Users/sheve/Desktop/Учебка/Практика/D/data/ND/var2.mat')

# data = data['mn']

# print(data)

# print('\nЗадание 7')

# corr\_matrix = np.zeros((6, 6))

# for i in range(0, 6):

# for j in range(0, 6):

# cor = np.corrcoef(data)

# corr\_matrix = cor

# np.set\_printoptions(precision=2)

# print(corr\_matrix)

# plt.plot(data[:, 2], data[:, 5], 'b.')

# plt.grid()

# plt.show()

**ПриложениеГ**

(обязательное)

**Листинг программного кода реализации алгоритма шифрования«Великий Шифр»**

# #script

# from memory import Key, Limit

# from random import randint,choice

# from re import findall

# # Омофоническое шифрование.

# keysCrypt = {

# 'А':Key[0:3], 'Б':Key[3:6], 'В':Key[6:9], 'Г':Key[9:12],

# 'Д':Key[12:15], 'Е':Key[15:18], 'Ё':Key[18:21], 'Ж':Key[21:24],

# 'З':Key[24:27], 'И':Key[27:30], 'Й':Key[30:33], 'К':Key[33:36],

# 'Л':Key[36:39], 'М':Key[39:42], 'Н':Key[42:45], 'О':Key[45:48],

# 'П':Key[48:51], 'Р':Key[51:54], 'С':Key[54:57], 'Т':Key[57:60],

# 'У':Key[60:63], 'Ф':Key[63:66], 'Х':Key[66:69], 'Ц':Key[69:72],

# 'Ч':Key[72:75], 'Ш':Key[75:78], 'Щ':Key[78:81], 'Ъ':Key[81:84],

# 'Ы':Key[84:87], 'Ь':Key[87:90], 'Э':Key[90:93], 'Ю':Key[93:96],

# 'Я':Key[96:99], ' ':Key[100:118]

# }

# # Кодирование.

# listWord =('ПОМОГИ','ЗАЧЕМ','ТУДА','ВОЙНА','НЕ','ВОЙТИ','ИЛИ','ЕЩЕ','ВХОД','ЧТО','ПРО','НИХ',

# 'КАК','АРБУЗ','ЕСЛИ','ОДИН','ВЫ','МНЕ','ИСПОЛЬЗОВАТЬ','ЕГО','СВОЙ','ОВОЩЬ',

# 'ВКЛЮЧИЛ','БЫЛ','БЫТЬ',' ЭТО','ЧТО','ОНИ','НЕТ','ДА','ИСТИНА','ЛОЖЬ','ЗВОНОК',

# 'ЧУВСТВОВАТЬ','БЛИЗКО', 'ОЧЕНЬ', 'КАКОЙ', 'АВТОМОБИЛЬ', 'ЛЮБОЙ', 'ДЕРЖИ',

# 'РАБОТАЙ','БЕГИ', 'НИКОГДА', 'НАЧИНАЙ', 'ДАЖЕ', 'СВЕТ', 'ЧЕМ',

# 'ПОСЛЕ', 'СТАВЬ',

# 'СТОП', 'СТАРЫЙ', 'СМОТРИ', 'ПЕРВЫЙ', 'МОЖЕТ', 'КЛАЛ', 'ДРУГОЙ', 'СЗАДИ',

# 'РЕЖЬ', 'ПОДЛО', 'УЛЫБАЙСЯ', 'НАШ', 'МНОГО', 'ОНО', 'ОН','ОНА', 'ЕЁ', 'ДОМ',

# 'ДЕРЖАТЬ', 'ДА', 'МЕСТО', 'НАЧАТЬ', 'НИЧЕГО', 'ГОД', 'МУЖЧИНА', 'ЖЕНЩИНА',

'ПОТОМУ', 'ТРИ', 'КАЖЕТСЯ', 'ЕСТЬ', 'ЖДАТЬ', 'НУЖНО', 'ПОСЛЕДНИЙ',

'ПОЗДНО', 'КОНЕЧНО', 'БОЛЬШОЙ', 'МАЛЕНЬКИЙ', 'СПЕРЕДИ', 'ДЕЙСТВИТЕЛЬНО',

'ИМЯ', 'ВСЕ', 'НОВЫЙ', 'ПАРЕНЬ','НУЖНО', 'ДОЛЖЕН', 'УБИТЬ', 'ТОЧКА',

'СТЕНА', 'ЧЕРНЫЙ', 'ШАГ', 'ВТОРОЙ', 'ЖИЗНЬ', 'БЫЛО', 'УПАСТЬ', 'СВОЙ',

'ДАЛЕКО', 'ПОКА', 'ЗА', 'ПОМОЧЬ', 'КОНЕЦ', 'ТЕ', 'ТО', 'ДОТЯНУТЬСЯ',

'ДЕВУШКА', 'УЛИЦА', 'РЯДОМ', 'НЕСКОЛЬКО', 'НОГИ', 'ПОКАЗАТЬ', 'ДОЛЖЕН', 'СТОЛ',

'ХОРОШО', 'ЕСТЬ', 'ХОРОШО', 'ТЕЛО', 'ТЕЛЕФОН', 'ДОБАВИТЬ', 'ВОДА', 'ОГОНЬ',

'ВНУТРИ', 'СЛОМАТЬ', 'КОГДА-НИБУДЬ', 'ВСТРЯХНУТЬ', 'ВСТРЕТИТЬСЯ', 'ОТЛИЧНО',

'РАЗУМ', 'ДОСТАТОЧНО', 'МИНУТА', 'СЛЕДОВАТЬ', 'АТАКОВАТЬ', 'МЕРТВ', 'ПОЧТИ')

# Стартовая позиция (конец шифра с омофонами).

position = 118

# Созданиесловаряс + позицией.

keysCode = {listWord[x]:Key[x + position] for x in range(len(listWord))}

# Шифр замены/шифр с заменой слогов.

listSyllables = ('АРЬ','ЕЦ','ВО','ВА','НА','СУ','ЖУ','ПРИ','НА','ТЫ','ПИ','БУ',

'МА','ПА','КА','ЗА','СЫ','ПО','ИВ','ИЗ','ПО','БЕЗ','МУЛЬТИ','ВО','ПРЕД','ОТ',

'НЕ', 'ПОД','!','?','.',',','@','#','$','%','\*','^','-','+','=','/',':',';','&','~')

# Стартовая позиция (конец кодирования).

position = len(listWord) + 118

# Созданиесловаряс + позицией.

keysSyllables = {listSyllables[x]:Key[x + position] for x in range(len(listSyllables))}

# Создание специальных символов (для редактирования текста).

listSpecial = ('<-','->','<+','+>')

# Стартовая позиция (конец шифра замены).

position = len(listWord) + len(listSyllables) + 118

# Созданиесловаряс + позицией.

keysSpecial = {listSpecial[x]:Key[x + position] for x in range(len(listSpecial))}

# Стартовая позиция (конец специальных символов).

position = len(listWord) + len(listSyllables) + len(listSpecial) + 118

# Создание символов-ловушек на все оставшиеся не занятые элементы памяти.

traps = tuple([Key[x] for x in range(position, Limit)])

# Удалитьненужныеэлементы.

del listWord, listSpecial, position

# Переключательрежимовшифрования.

cryptMode = input("[1]Зашифровать|[2]Разшифровать: ").upper()

if cryptMode not in ['1','2']:

print("Ошибка: режим не найден")

raise SystemExit

# Сообщение для шифрования/расшифрования.

startMessage = input("Напишитесообщение: ").upper()

# Функция возвращающая триады чисел.

def regular(text):

return findall(r"[0-9]{3}", text)

# Главнаяфункция.

def encryptDecrypt(mode, message, final = "", string = ""):

# Если переключатель равен шифрованию, тогда разделить сообщение по словам.

if cryptMode == '1':

secondText = findall(r"[^\s]+", message)

# Удалить ненужные элементы.

del message

# Внедрение в код специальных символов.

for indexWord in range(len(secondText)):

if secondText[indexWord] in keysSpecial:

secondText[indexWord] = keysSpecial[secondText[indexWord]]

# Кодированиесловначисла.

for indexWord in range(len(secondText)):

if secondText[indexWord] in keysCode:

secondText[indexWord] = keysCode[secondText[indexWord]]

# Заменаслоговначисла.

for indexWord in range(len(secondText)):

for syllable in keysSyllables:

if syllable in secondText[indexWord]:

secondText[indexWord] = secondText[indexWord].replace(syllable,keysSyllables[syllable])

# Разделение всех слов на символы.

for indexWord in range(len(secondText)):

secondText[indexWord] = list(secondText[indexWord])

# Добавление в конец каждого списка пробел.

for indexWord in range(len(secondText)):

secondText[indexWord].append(' ')

# Шифрование всех символов на числа при помощи омофоноф.

for indexWord in range(len(secondText)):

for indexSymbol in range(len(secondText[indexWord])):

symbol = secondText[indexWord][indexSymbol]

if symbol in keysCrypt:

length = len(keysCrypt[symbol])

secondText[indexWord][indexSymbol] = keysCrypt[symbol][randint(0, length - 1)]

# Соединение всех числе в одну строку.

for word in secondText:

string += "".join(word)

# Разделениечиселпотриадам.

finalList = list(regular(string))

# Внедрение в сообщение ложных символов.

for indexList in range(len(finalList)):

randSwitch = randint(0,2); randPosition = randint(0,len(finalList))

if not randSwitch: finalList.insert(randPosition,choice(traps))

# Занесение в переменную final зашифрованный текст.

for word in finalList:

final += "".join(word)

# Вернуть зашифрованное сообщение.

return ".".join(regular(final))

# Если переключатель равен расшифрованию, тогда сделать перебор сообщения через функцию, возвращающая триады чисел.

else:

for symbolText in regular(message):

# Перебор всех специальных символов.

forelementinkeysSpecial:

if symbolText == keysSpecial[element]: final += element

# Переборвсехкодов.

for word in keysCode:

if symbolText == keysCode[word]: final += word

# Перебор всех символов и слогов шифра.

for syllable in keysSyllables:

if symbolText == keysSyllables[syllable]: final += syllable

# Переборвсехомофонов.

for symbol in keysCrypt:

if symbolText in keysCrypt[symbol]: final += symbol

# Разделить слова между собой в список.

listWord = findall(r"[^\s]+",final)

# Переборспискаслов.

for \_ in range(len(listWord)):

# Если специальный символ будет являться словом в списке, тогда сделать действия.

for element in keysSpecial:

if element in listWord:

# Если найдено слово ‘x’.<Typex>-’, тогда удалить предыдущее слово и этот символ.

if element == '<-':

del listWord[listWord.index(element) - 1]

listWord.remove(element)

elif element == '->':

del listWord[listWord.index(element) + 1]

listWord.remove(element)

elif element == '<+':

listWord[listWord.index(element)] = listWord[listWord.index(element) - 1]

elif element == '+>':

listWord[listWord.index(element)] = listWord[listWord.index(element) + 1]

else: pass

# Вернуть расшифрованное сообщение.

return " ".join(listWord)

# Выводрезультата.

print("Финальноесообщение:", encryptDecrypt(cryptMode, startMessage))

#memory

Limit = 350;

Key = (

'275', '078', '276', '230', '343', '127', '006', '325', '307', '102',

'334', '185', '004', '002', '008', '283', '277', '260', '256', '305',

'300', '143', '159', '248', '160', '309', '104', '222', '136', '317',

'264', '053', '218', '137', '177', '315', '301', '244', '040', '072',

'023', '182', '323', '154', '076', '048', '213', '330', '109','164',

'257', '179', '043', '166', '207', '240', '220', '205', '228', '073',

'017', '093', '186', '027', '157', '080', '009', '195', '289', '278',

'116', '238', '028', '271', '058', '001', '132', '236', '044', '029',

'282', '339', '131', '069', '100', '107', '255', '135', '226', '198',

'335', '035', '187', '119', '293', '133', '270', '347', '097', '273',

'068', '250', '311', '140', '212', '120', '024', '303', '188', '103',

'209', '114', '034', '267', '061', '184', '261', '225',

'349', '167', '059', '106', '272', '168', '326', '165', '269', '279',

'019', '247', '087', '156', '239', '088', '092', '026', '071', '304',

'241', '180', '306', '284', '310', '065', '021', '234', '162', '202',

'060', '047', '039', '318', '321', '152', '117', '324', '067', '348',

'031', '308', '265', '070', '181', '077', '217', '041', '032', '124',

'125', '290', '235', '037', '231', '003', '342', '158', '020', '322',

'216', '091', '176', '079', '199', '259', '129', '138', '113', '099',

'312', '215', '115', '145', '122', '193', '254', '314', '153', '344',

'123', '340', '292', '101', '083', '286', '262', '183', '121', '237',

'242', '249', '246', '139', '018', '042', '295', '178', '245', '022',

'148', '163', '052', '233', '189', '229', '090', '000', '341', '015',

'346', '336', '082', '345', '281', '036', '005', '089', '038', '064',

'141', '046', '151', '062', '266', '243', '010', '287', '066', '149',

Приложение Д

(справочное)

**Библиографический список**

1. Обзор Spyder[электронный ресурс] // Официальный сайт Spyder. URL: <https://www.spyder-ide.org/>(дата обращения 16.07.2020).
2. Python [электронный ресурс] // Википедия. Свободная энциклопедия. URL: <https://ru.wikipedia.org/wiki/Python>(дата обращения 16.07.2020).
3. История криптографии[электронный ресурс] // Сборник статьей. URL: <https://kod.ru/istoriia-kriptoghrafii-ot-stieghanoghrafii-do-slozhnykh-alghoritmov-chast-iii/>(дата обращения 16.07.2020).