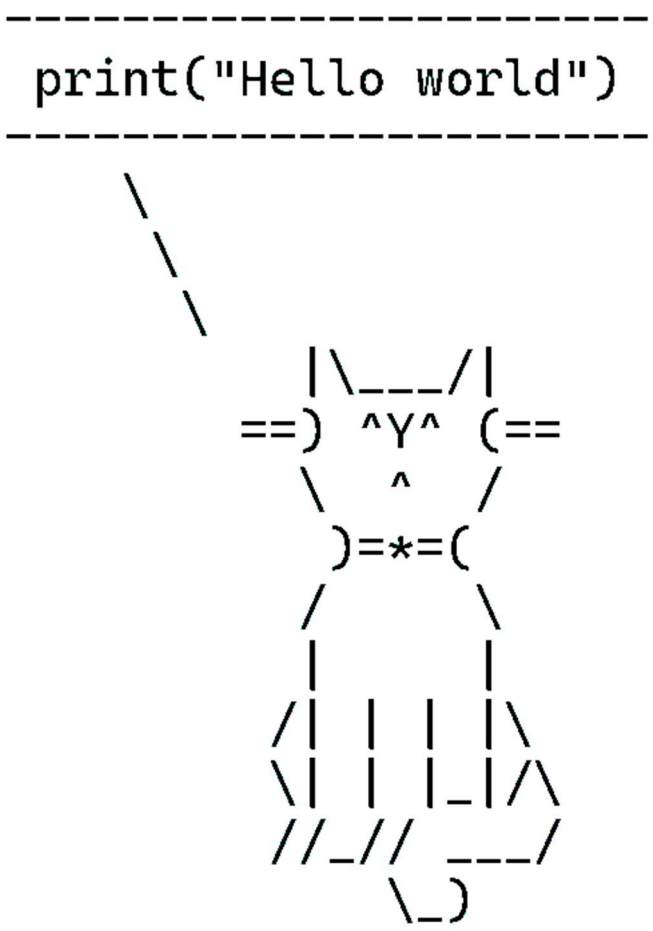
ИТМО

М. Я. Афанасьев, А. А. Крылова, С. А. Шорохов, Федосов Ю. В.

ПРОГРАММИРОВАНИЕ НА PYTHON



Санкт-Петербург
2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

М. Я. Афанасьев, А. А. Крылова, С. А. Шорохов,
Федосов Ю. В.

ПРОГРАММИРОВАНИЕ НА PYTHON

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлениям подготовки
12.03.01, 13.03.02, 15.03.04, 15.03.06, 24.03.02, 27.03.04
в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург
2025

Афанасьев М. Я., Крылова А. А., Шорохов С. А., Федосов Ю. В.,
Программирование на Python – СПб: Университет ИТМО, 2025. – 64 с.

Рецензент(ы):
Ловлин Сергей Юрьевич, кандидат технических наук, доцент (квалификационная категория "ординарный доцент") факультета систем управления и робототехники, Университета ИТМО.

Методические указания разработаны для студентов, обучающихся по направлениям, связанным с программированием и инженерией, и предназначены для углубленного освоения языка Python. Пособие охватывает ключевые аспекты, необходимые для формирования базовых и продвинутых навыков программирования, таких как работа с управляющими символами, анализ данных, создание графических интерфейсов и алгоритмизация.

Оглавление

Введение

Дисциплина «Программирование на Python» является одним из ключевых курсов, направленных на освоение основ программирования и формирования практических навыков разработки программного обеспечения. Язык Python широко используется в различных областях, таких как веб-разработка, научные исследования, анализ данных, искусственный интеллект и автоматизация процессов, благодаря своей простоте, читаемости и мощности.

Настоящий лабораторный практикум предназначен для студентов, изучающих программирование на Python в рамках практических занятий. В них представлены основные теоретические аспекты языка, а также практические задания, которые помогут закрепить изученный материал и развить навыки написания кода, работы с алгоритмами, использования библиотек и решения реальных задач.

Практикум включает задания различного уровня сложности, начиная от базовых конструкций языка (переменные, условные операторы, циклы) до более продвинутых тем, таких как работа с файлами, обработка данных, использование внешних библиотек и создание простых программных проектов. Выполнение этих заданий поможет студентам не только освоить базовые конструкции языка, но и научиться разрабатывать эффективные и структурированные программы.

Методические указания также содержат рекомендации по выполнению практических заданий и подготовке отчетов. Основной целью курса является развитие у студентов навыков логического мышления, анализа и решения задач с использованием языка программирования Python, что будет полезно для их дальнейшего обучения и профессиональной деятельности.

Лабораторный практикум предназначен для студентов бакалавриата, обучающихся по направлениям: 12.03.01 Приборостроение, 13.03.01 Теплоэнергетика и теплотехника, 13.03.02 Электроэнергетика и электротехника, 15.03.04 Автоматизация технологических процессов и производств, 15.03.06 Мехатроника и робототехника, 24.03.02 Системы управления движением и навигация, 27.03.04 Управление в технических системах.

При выполнении данного лабораторного практикума в процессе освоения обучающимися дисциплины «Программирование на Python» формируются компетенции, связанные с исследованием и применением интеллектуальных систем для различных предметных областей, а также использованием методов и инструментов инженерии знаний. В частности, обучающиеся закрепляют знания основ синтаксиса языка программирования Python и продвинутых техник программирования; приобретают умения программировать в объектно-ориентированной парадигме, в том числе web-приложений; приобретают практические навыки работы с функциями и данными, работы с прикладными библиотеками Python и командной работы с системой контроля версий git.

Программа лабораторного практикума

Практикум включает шесть лабораторных работ.

Лабораторная работа 1. Управляющие символы ANSI и вывод в консоль

Цель работы – освоение методов использования escape-символов для управления выводом информации в консоли. Студенты научатся генерировать графические элементы, такие как изображения флагов, узоры, графики функций и диаграммы, в текстовом формате с помощью символов и управляющих последовательностей. В процессе выполнения заданий студенты также познакомятся с методами обработки и визуализации данных на основе файлов с числовыми последовательностями. Дополнительно студенты смогут применить знания по управлению консольным выводом для создания простой анимации, что позволит лучше понять принципы работы с консолью и основами программирования графических интерфейсов в текстовой среде.

Лабораторная работа 2. Работа с табличными форматами

Цель работы – реализовать анализ данных из файлов с книгами: подсчитать записи с длинными названиями, выполнить поиск книг по автору с ограничением на количество результатов, сгенерировать библиографические ссылки для случайных записей и сохранить их в текстовом формате, а также распарсить XML-файл валют, выполняя приведение типов по необходимости.

Лабораторная работа 3. Работа в Tkinter

Цель работы – получить практические навыки о программировании пользовательского интерфейса на примере библиотеки tkinter, а также о генерации численно-буквенных последовательностей с использованием библиотеки random.

Лабораторная работа 4. Задача о рюкзаке

Цель работы – получить практические навыки в реализации алгоритмов по их формальному и схематическому описанию.

Лабораторная работа 5. Работа с регулярными выражениями

Цель работы – получить практические навыки использования регулярных выражения для поиска необходимой информации в файлах с различным содержимым.

Лабораторная работа 6. Объектно-ориентированное программирование

Цель работы – освоить основные принципы объектно-ориентированного программирования (ООП) в Python, включая создание классов и объектов, применение принципов инкапсуляции, наследования и полиморфизма, а также изучение особенностей множественного наследования и методов для управления доступом к данным внутри классов.

Требования к оформлению отчета по лабораторным работам

Общие требования

1) Отчет выполняется в виде самодостаточного документа. Материал, изложенный в отчете, должен быть понятен без дополнительных комментариев со стороны исполнителей.
2) Отчет выполняется как текстовый документ в соответствии с ГОСТ 2.105-95 и предоставляется в электронном виде.
3) В отчёте должен быть представлен листинг программы на языке Python, а также обязательная ссылка на репозиторий с проектом на ресурсе `github.com`. Строки листинга должны быть пронумерованы.
4) Листы отчета должны быть пронумерованы, кроме титульного листа, который считается первым.
5) Если отчет содержит большое количество листов, рекомендуется добавлять лист с содержанием отчета (разделы и номера листов).

Содержание отчета

Отчет должен содержать следующие разделы:

1) Титульный лист.
2) Цель работы.
3) Задачи, решаемые в работе.
4) Теоретическая часть.
5) Экспериментальная часть.
6) Выводы.

Шаблон отчета представлен в приложении Д. Исходные коды программ можно найти по адресу https://github.com/ITMOPython-2022.

Цель работы

В данном разделе должна быть сформулирована цель работы. Допускается копирование цели работы из методических указаний.

Задачи, решаемые в работе

В данном разделе должны быть сформулированы задачи, которые необходимо решить в процессе выполнения лабораторной работы для достижения поставленной цели.

Теоретическая часть

Теоретическая часть отчета по лабораторной работе должна включать в себя всю необходимую информацию о предметной области, к которой относятся

описание работы программных компонентов, используемых при ее выполнении, описание алгоритмов и программных библиотек и т. п.

Данный раздел не является обязательным, однако настоятельно рекомендуется оставлять его в отчете, так как он может быть использован как план ответа при защите лабораторной работы. В случае, если теоретическая части отчета по лабораторной работе содержит листинги программного кода, таблицы или рисунки, то они должны быть оформлены по правилам, описанным далее.

Экспериментальная часть

Часть отчета, описывающая экспериментальную или практическую часть лабораторной работы, является обязательной и не может быть опущена. Данная часть отражает персональные результаты обучающегося, полученные им при выполнении лабораторной работы. Этими результатами являются листинги написанного программного кода, таблицы и изображения, полученные в процессе решения сформулированных задач. Исходные данные, использованные обучающимся для практической проверки реализованных в ходе практической работы программных компонентов, также должны быть представлены в экспериментальной части.

Данные по выполненной лабораторной работе можно предоставить отдельно, в том числе в виде ссылки на Интернет-ресурс. При изучении ранее выполненных работ обучающимся рекомендуется самостоятельно переписать всю программу в том виде, в каком она была написана, с целью лучшего понимания алгоритма программы.

Если студент не завершил цикл выполнения программы, должно быть проведено ее окончательное редактирование. В экспериментальную часть отчета по лабораторной работе необходимо включить пояснения и комментарии к написанному программному коду.

Выводы

Выводы должны быть кратко изложены в виде списка и отражать наиболее важные аспекты лабораторной работы. В конце отчета студент должен привести список использованной при подготовке отчета и процитированной литературы (в том числе ссылки на стандарты, руководства пользователя и прочую техническую документацию). Текст отчета по лабораторной работе может являться планом ответа при защите лабораторной работы. Прямое чтение текста отчета не допускается.

Правила оформления текста отчета по лабораторной работе

Отчет должен быть представлен в формате PDF. Поля текста могут быть выбраны произвольно, например, 25 мм по всем сторонам печатного листа. Рекомендуется при оформлении отчета использовать шрифт чёрного цвета с засечками (например, DejaVu Serif) высотой 14 пунктов с полуторным интервалом между строками. Обязательным требованием при оформлении текста отчёта по лабораторной работе является его выравнивание по ширине страницы и использование автоматических переносов.

На рисунках в отчете могут быть представлены блок-схемы, графики, снимки экранов виртуального окружения, а также прочая графическая информация. Рисунки должны быть четкими и легкочитаемыми. Если рисунок является снимком экрана, то настоятельно рекомендуется сохранять исходное в формате без сжатия (например, PNG) или копировать снимок экрана из буфера обмена непосредственно в программу, в которой редактируется текст отчета (например, LibreOffice). Графики и диаграммы рекомендуется создавать в векторных форматах. Ниже представлен пример оформления рисунка.

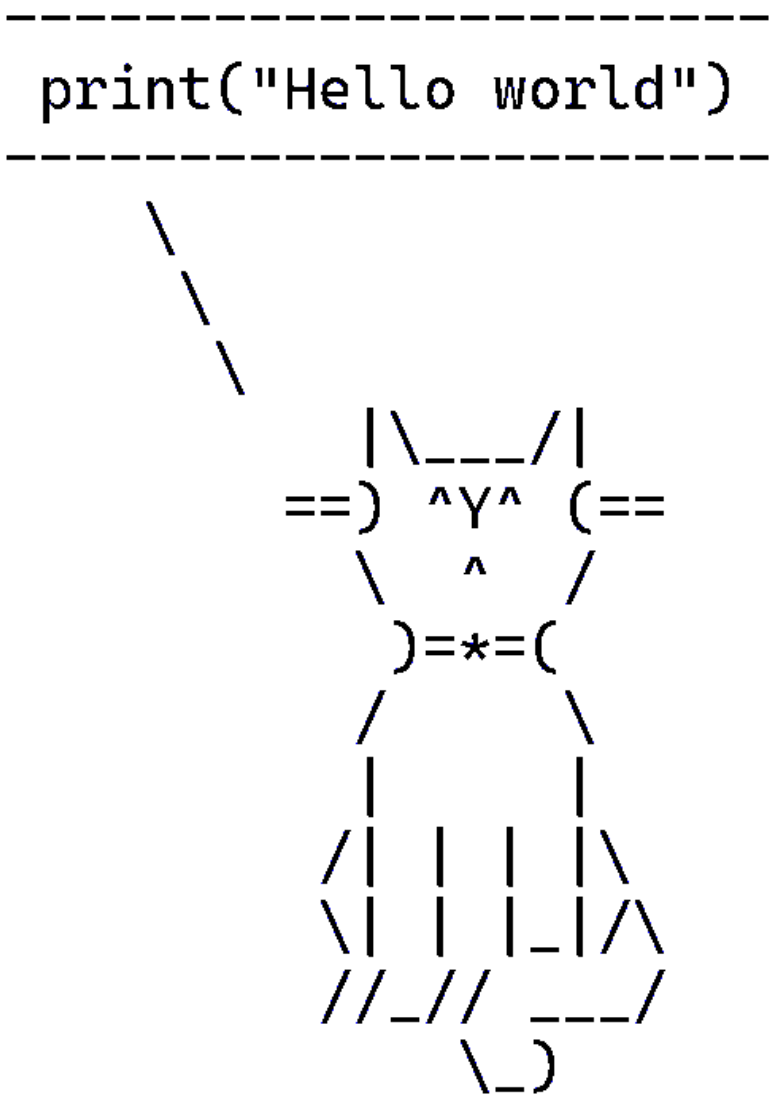


Рисунок X – Пример оформления рисунка

Листинги программного кода должны оформляться как скриншоты окна редактора, либо непосредственно в тексте отчета. В обоих случаях должны быть соблюдены следующие правила:

1) Необходимо использовать моноширинные шрифты.
2) Строки программы должны быть пронумерованы, нумерация должна начинаться с единицы.

3) Должна быть включена «подсветка» синтаксических конструкций языка программирования Python.

Исходные коды должны быть снабжены комментариями. На рисунки и листинги обязательно должны присутствовать ссылки в тексте. Например "На рисунке 1 изображен ... " или "Исходный код программы представлен в листинге 2 ...". Листинги, таблицы и рисунки должны быть снабжены подписями, отражающими их содержание.

Защита лабораторной работы

Во время защиты лабораторной работы обучающемуся следует ясно и чётко изложить цель работы и основные решённые задачи, а также ответить на дополнительные вопросы, заданные преподавателем в процессе защиты отчёта. Требования к защите отчета представлены в таблице 1.

Таблица 1 – Таблица оценивания защиты лабораторной работы

№ п/п	**Требования**	**Оценка (уровень)**		
		высокий	средний	низкий
1	Уровень оформления отчета	2	1	0,5
2	Навыки устного представления результатов работы	2	1	0,5
3	Понимание (воспроизведение) написанного программного кода	2	1	0,5
4	Правильность и полнота понимания достигнутого результата	2	1	0,5
5	Качество ответов на дополнительные вопросы	2	1	0,5
	Итого баллов:	**10**	**5**	**2,5**

Лабораторная работа №1

Управляющие символы ANSI и вывод в консоль

Введение

Командная строка, консоль или терминал – это неотъемлемый инструмент программиста. Он присутствует в операционной системе независимо от того, пользуетесь ли вы Windows, Linux, Mac OS или чем-то еще. Терминал представляет собой приложение, которое позволяет ввести текстовую команду, после чего обрабатывает ее и выводит текстовый результат. Такой режим работы также называют REPL (от англ. Read-Eval-Print Loop), что означает цикл «чтение-вычисление-запись». Фактически терминал ожидает от вас текстовые команды, написанные на специальном языке программирования терминала. Например, для Windows – это batch, для Linux и Mac – bash или shell. Кстати, в режиме REPL может работать и интерпретатор Python, ожидая команды на языке Python. Ниже приведен пример ввода команды **echo** в терминал, а также команда запуска программы **test.py** на Python через терминал (см. рисунок 1.1).

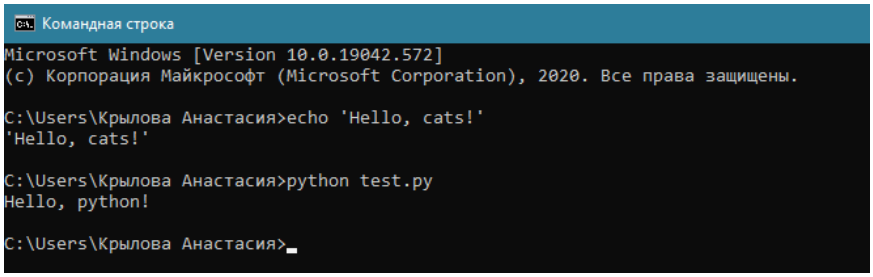


Рисунок 1.1 – Пример работы с терминалом в Windows

Процедура вывода в терминал является частью работы множества программ. Ее используют для вывода прогресса выполнения, отображения ошибок, текстовых сообщений отладки, упрощенной визуализации данных и т. д. В языке Python, чтобы вывести текст в консоль используется функция `print()`.

```
print('Knock, knock, Neo.')
print(‘The matrix has you..')
```

Полезные ссылки:

- Флаги разных стран: https://www.worldometers.info/geography/flags-of-the-world/
- Об escape-последовательностях: https://docs-python.ru/tutorial/strokovye-bajtovye-literaly/escape-posledovatelnosti-python/
- И ещё: https://dvmn.org/encyclopedia/python_strings/ansi-codes/
- Примеры: https://www.lihaoyi.com/post/BuildyourownCommandLinewithANSIescapecodes.html

Описание лабораторной работы

Цель работы

Освоение методов использования escape-символов для управления выводом информации в консоли. Студенты научатся генерировать графические элементы, такие как изображения флагов, узоры, графики функций и диаграммы, в текстовом формате с помощью символов и управляющих последовательностей. В процессе выполнения заданий студенты также познакомятся с методами обработки и визуализации данных на основе файлов с числовыми последовательностями. Дополнительно студенты смогут применить знания по управлению консольным выводом для создания простой анимации, что позволит лучше понять принципы работы с консолью и основами программирования графических интерфейсов в текстовой среде.

Указания к выполнению работы

Работа состоит из двух частей. Первая часть отведена под проектирование внешнего вида пользовательского интерфейса программы с размещением необходимых элементов. Вторая часть заключается в реализации полезной функции программы. Имеется дополнительное задание по расширению функционала написанной программы, которое выполняется по желанию.

Оформление отчета

Отчёт должен быть составлен в соответствии с требованиями, представленными во введении.

Ход работы

Для начала определяем несколько констант, описывающих изменение цвета текста в консоли

```
RED = '\u001b[41m'
BLUE = '\u001b[44m'
WHITE = '\u001b[47m'
```

Затем определяем константу, которая отменяет изменение цвета и возвращает значения цветов текста в консоли к изначальным

```
END = '\u001b[0m'
```

После этого реализуем простой цикл для отображения в консоли флага Чехии в виде псевдографики:

```
for i in range(6):
    if i < 3:
```

```
            print(f'{BLUE}{"  " * (2 * i + 2)}{WHITE}{"  " * (14 - 2 * i)}{END}')
        else:
            print(f'{BLUE}{"  " * (12 - 2 * i)}{RED}{"  " * (4 + 2 * i)}{END}')
```

Далее реализуем отрисовку графика кубической функции в первом квадранте числовой оси:

```
plot_list = [[0 for i in range(10)] for i in range(10)]
result = [0 for i in range(10)]

for i in range(10):
    result[i] = i ** 3

step = round(abs(result[0] - result[9]) / 9, 2)
print(step)

for i in range(10):
    for j in range(10):
        if j == 0:
            plot_list[i][j] = step * (8-i) + step

for i in range(9):
    for j in range(10):
        if abs(plot_list[i][0] - result[9 - j]) < step:
            for k in range(9):
                if 8 - k == j:
                    plot_list[i][k+1] = 1

for i in range(9):
    line = ''
    for j in range(10):
        if j == 0:
            line += '\t' + str(int(plot_list[i][j])) + '\t'
        if plot_list[i][j] == 0:
            line += '--'
        if plot_list[i][j] == 1:
            line += '!!'
    print(line)
print('\t0\t1 2 3 4 5 6 7 8 9')
```

Пример работы со сложными псевдографическими изображениями и анимацией представлен в приложении А.

Для работы с числовой последовательностью используется следующая заготовка, реализующая открытие файла на чтение и отображения на экране содержимого файла:

```
file = open('sequence.txt', 'r')
list = []

for number in file:
    list.append(float(number))
file.close()

print(list)
```

Задание и варианты

1) Сгенерировать при помощи escape-символов в консоли изображение флага в соответствии с вариантом (столбец «Страна»).
2) Сгенерировать в консоли повторяющийся узор (столбец «Узор»).
3) Сгенерировать в консоли график функции (1 квадрант) при помощи escape-символов, минимум 9 строк в высоту (столбец «Функция»).
4) Используя прилагаемый файл с числовой последовательностью sequence.txt, вывести диаграмму процентного соотношения согласно варианту.

Дополнительное задание

Используя функцию очищения консольного вывода (`os.system("cls") или os.system("clear")`), реализовать анимацию из 2-3 кадров.

Варианты задания представлены в таблице 1.1

Таблица 1.1 – Варианты задания для лабораторной работы № 1

Вариант	Страна	Узор	Функция	Условие
1	Франция	a	$y=x^2$	Количество чисел меньше и больше 0
2	Бангладеш	b	y=2x+3	Сумма по модулю первых 125 чисел и вторых 125 чисел
3	Нидерланды	c	y=2x	Сумма чисел по модулю, стоящих на чётных и нечётных позициях
4	Польша	d	$y=x^{0.5}$	Среднее по модулю первых 125 и вторых 125 чисел
5	Литва	e	y=|x|	Среднее по модулю чисел, стоящих на чётных и нечётных позициях

6	Таиланд	f	y = 1 / x	Числа больше 5 и меньше 5, отрицательные отбросить
7	Япония	g	y = 3x	Числа больше -5 и меньше -5, положительные отбросить
8	Бенин	h	y = x + 1	Числа от 0 до 5 и числа от 0 до -5, остальные отбросить
9	Финляндия	i	y = x / 2	Числа от 5 до 10 и числа от -5 до -10, остальные отбросить
10	Швейцария	j	y = x / 3	Числа от -3 до 3 и остальные

Варианты узоров представлены на рисунке 1.2

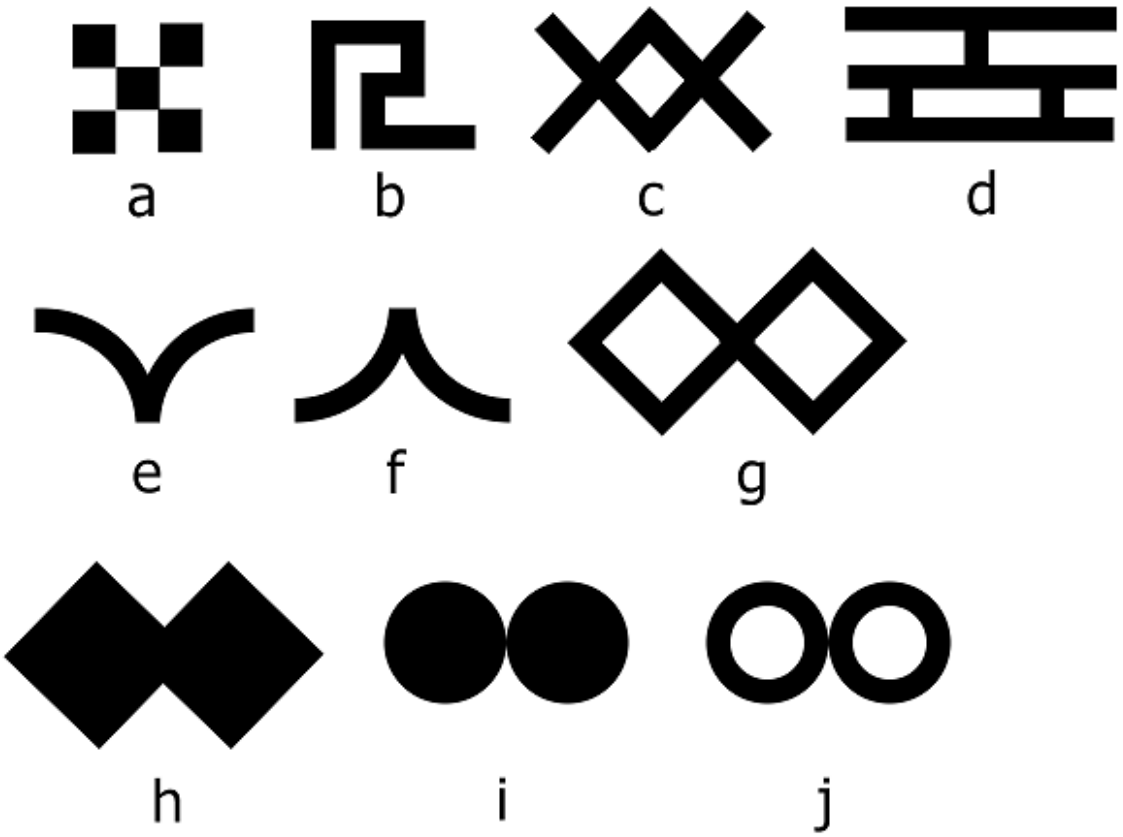


Рисунок 1.2 – Варианты узоров

Контрольные вопросы

1) Что такое escape-последовательности в Python, и для чего они используются?
2) Какую escape-последовательность нужно использовать для добавления новой строки в текст? Приведите пример использования.
3) Как с помощью escape-последовательности вставить символ табуляции в строку? Приведите пример строки с использованием этой последовательности.
4) Какая escape-последовательность позволяет вставить в текст обратную косую черту (\)?

Лабораторная работа №2

Работа с табличными форматами

Введение

Табличные форматы данных – это стандарты представления структурированной информации, которые позволяют упорядочить и хранить данные в виде таблиц или иерархических структур. Среди наиболее популярных форматов для работы с табличными данными выделяются CSV, XML и JSON. Каждый из этих форматов имеет свои особенности, применимость и преимущества в различных областях. Рассмотрим их подробнее.

CSV (значения, разделённые запятыми) – это простой текстовый формат, предназначенный для хранения табличных данных. В этом формате каждая строка представляет собой одну запись, а значения в ней разделены запятыми (или другим разделителем, таким как точка с запятой или табуляция). Строки разделяются переносом строки, что делает CSV чрезвычайно компактным и легким для чтения и обработки.

Основные характеристики CSV:

- Простота и лёгкость использования: CSV-файл представляет собой обычный текст, который можно открыть и редактировать в любом текстовом редакторе.
- Широкая поддержка: CSV поддерживается большинством программ, работающих с табличными данными, такими как Microsoft Excel, Google Sheets и практически все языки программирования.
- Отсутствие вложенной структуры: CSV подходит для представления только плоской, двумерной таблицы данных, где каждая строка представляет отдельную запись, а столбцы — отдельные поля.

Преимущества CSV:

- Простота и универсальность: CSV-файлы легко читаются и обрабатываются большинством инструментов и программ.
- Лёгкость передачи и хранения: благодаря компактному размеру, CSV-файлы занимают мало места и быстро передаются.

Недостатки CSV:

- Нет поддержки вложенности: формат не поддерживает иерархическую или многомерную структуру данных.
- Нет типизации данных: все данные в CSV представляют собой строки, что требует дополнительных преобразований для работы с числами, датами и другими типами.

XML (расширяемый язык разметки) – это формат данных, предназначенный для хранения и обмена структурированной информацией. В отличие от CSV, XML поддерживает иерархическую структуру данных и подходит для сложных

структур, где данные содержат вложенные элементы. XML-данные представлены в виде дерева, где каждый узел является элементом с атрибутами и значениями.

Основные характеристики XML:

- Гибкость структуры: XML поддерживает сложные структуры данных, что позволяет описывать любые типы данных, от простых до сложных иерархий.
- Самоописание: XML позволяет добавлять метаданные (описания) для каждого элемента, что помогает понять структуру и контекст данных.
- Широкая поддержка в различных системах: XML используется во многих веб-сервисах и интеграционных решениях, поддерживается практически всеми языками программирования.

Преимущества XML:

- Поддержка иерархии и вложенности: XML позволяет представлять сложные многомерные структуры данных.
- Читабельность для человека: данные в XML легко читаются благодаря меткам, описывающим каждый элемент.
- Расширяемость: XML позволяет использовать атрибуты и добавлять новые элементы по мере необходимости.

Недостатки XML:

- Избыточность: XML-файлы часто занимают много места из-за необходимости включать теги для каждого элемента и атрибута.
- Сложность обработки: считывание и запись XML требует больше ресурсов, чем работа с простыми текстовыми файлами.
- Отсутствие встроенной поддержки типизации: хотя XML может содержать сложные данные, типы этих данных (строки, числа) нужно обрабатывать отдельно.

JSON (JavaScript Object Notation) – это формат, изначально разработанный для обмена данными в JavaScript, но ныне широко используемый во многих языках и приложениях. JSON используется для представления данных в виде пары «ключ-значение» и поддерживает вложенные структуры, подобные структурам в XML, что делает его популярным выбором для API и веб-сервисов.

Основные характеристики JSON:

- Легковесность и простота: JSON-файлы обычно компактнее, чем XML, что упрощает их использование и передачу.
- Чёткая структура данных: JSON поддерживает сложные объекты с вложенностью, что делает его подходящим для хранения любых типов данных.
- Широкая поддержка в веб-приложениях: JSON является стандартом для обмена данными в большинстве веб-приложений и используется практически всеми современными языками программирования.

Преимущества JSON:

- Простота и компактность: JSON легче и быстрее обрабатывается по сравнению с XML.
- Хорошая поддержка типизации: JSON может содержать строки, числа, массивы и объекты, что позволяет легко работать с данными различных типов.
- Широкое распространение и поддержка в веб-разработке: JSON стал стандартом обмена данными между сервером и клиентом в веб-приложениях.

Недостатки JSON:

- Ограничения в типах данных: JSON не поддерживает сложные типы данных, такие как даты или двоичные данные, без дополнительных преобразований.
- Ограниченная поддержка комментариев: В JSON нельзя добавлять комментарии, что может затруднить его использование для хранения самоописательных данных.
- Ограниченная поддержка схем: В отличие от XML, JSON не имеет встроенной схемы для валидации данных, хотя существуют инструменты для валидации JSON (например, JSON Schema).

Каждый из рассмотренных форматов данных – CSV, XML и JSON – имеет свои преимущества и недостатки в зависимости от области применения. CSV удобен для представления простых таблиц, XML поддерживает сложные структуры данных с иерархией, а JSON стал стандартом для обмена данными в веб-приложениях и поддерживает структурированные данные с меньшим объёмом избыточной информации по сравнению с XML.

Полезные ссылки:

- О формате CSV: https://blog.skillfactory.ru/glossary/csv/
- О библиотеке xml.dom: https://docs.python.org/3/library/xml.dom.minidom.html
- О формате XML: https://help.reg.ru/support/hosting/razmeshcheniye-sayta-otobrazheniye-v-brauzere/chto-takoye-format-xml

Описание лабораторной работы

Цель работы

Реализовать анализ данных из файлов с данными о книгах: подсчитать записи с длинными названиями, выполнить поиск книг по автору с ограничением на количество результатов, сгенерировать библиографические ссылки для случайных записей и сохранить их в текстовом формате, а также распарсить XML-файл валют, выполняя приведение типов по необходимости.

Указания к выполнению работы

Работа состоит из четырех частей. В первой части производится фильтрация таблицы CSV по определенному параметру, во второй части реализуется поиск по таблице, в третьей части реализуется генерация отчета по строке таблицы CSV, в четвертой части реализуется парсинг файла XML. Имеется дополнительное задание по расширению функционала написанной программы, которое выполняется по желанию.

Оформление отчета

Отчёт должен быть составлен в соответствии с требованиями, представленными во введении.

Ход работы

Реализуем простой поиск по таблице CSV из файла **`civic.csv`**. Код получает запрос от пользователя и выводит результаты поиска в файл.

```
from csv import reader

output = open('result.txt', 'w')

while True:
    flag = 0
    search = input('Введите запрос: ')

    if search == '0':
        break

    with open('civic.csv', 'r', encoding='windows-1251') as csvfile:
        table = reader(csvfile, delimiter=';')
        for row in table:
            lower_case = row[2].lower()
            index = lower_case.find(search.lower())
            if index != -1:
                print(row[2])
                output.write(
```

```
                    f'{row[2]}. Цена {row[8]} руб. S/n {row[18]}\n'
                )
                flag += 1

        if flag == 0:
            print('Ничего не найдено.')
        else:
            print(f'Найдено {flag} результатов.')

output.close()
```

Далее рассмотрим код, открывающий файл в формате XML и осуществляющий грамматический разбор этого файла.

```
import xml.dom.minidom as minidom


xml_file = open('books.xml', 'r')
xml_data = xml_file.read()

dom = minidom.parseString(xml_data)
dom.normalize()

elements = dom.getElementsByTagName('book')
books_dict = {}

for node in elements:
    for child in node.childNodes:
        if child.nodeType == 1:
            if child.tagName == 'title':
                if child.firstChild.nodeType == 3:
                    title = child.firstChild.data
            if child.tagName == 'price':
                if child.firstChild.nodeType == 3:
                    price = float(child.firstChild.data)
    books_dict[title] = price

    if node.getAttribute('id') == 'bk106':

print(node.getElementsByTagName('title')[0].firstChild.data)

for key in books_dict.keys():
    print(key, books_dict[key])
```

```
print(books_dict)

xml_file.close()
```

Пример, реализующий более сложный способ поиска в табличном файле, представлен в приложении Б.

Задание и варианты

Используя приложенный файл `books.csv` ИЛИ `books-en.csv`, выполнить следующее:

- Вывести количество записей, у которых в поле Название строка длиннее 30 символов.
- Реализовать поиск книги по автору, использовать ограничение на выдачу в зависимости от варианта.
- Реализовать генератор библиографических ссылок вида <автор>. <название> - <год> для 20 записей. Записи выбрать произвольно. Список сохраняется как отдельный файл текстового формата с нумерацией строк.

Используя приложенный файл `currency.xml`, выполнить следующее:

Распарсить файл и извлечь данные согласно варианту. Выполнить приведения типов по необходимости.

Дополнительное задание

- Вывести перечень всех тегов без повторений (для books-en.csv - перечень издательств без повторений).
- Самые популярные 20 книг.

Таблица 2.1 – Варианты задания для лабораторной работы № 2

Варианты	**Ограничения books.csv**	**Ограничения books-en.csv**	**XML**
1	До 150 рублей	До 150 рублей	Словарь `"Name - Value"`
2	До 2016 года	1991 и 1996 года	Два отдельных списка `Name` и `Value`
3	Только 2014, 2016 и 2017 года	До 1990 года	Список `Name`, но только для валют с `Nominal=1`
4	До 200 рублей	До 200 рублей	Словарь `"NumCode - CharCode"`
5	Нет	Нет	Два отдельных списка `CharCode` и `Value`
6	От 150 рублей	От 150 рублей	Средний показатель `Value`
7	От 2016 до 2018 года	От 1991 до 1995	Список `CharCode`, но только для валют с `Nominal=10` или

			`Nominal=100`
8	Только 2015 и 2018 года	books-en - от 1997 до 2000	Словарь `"CharCode – Nominal"`
9	От 200 рублей	От 200 рублей	Два отдельных списка `NumCode` и `CharCode`
10	От 2018 года	От 2000 года	Словарь `"Name – CharCode"`

Контрольные вопросы

1) В чем заключается основное отличие формата CSV от форматов XML и JSON с точки зрения структуры данных?
2) Какие преимущества и недостатки имеет формат XML при работе с вложенными данными?
3) Почему JSON стал стандартом для обмена данными в веб-приложениях, и какие особенности делают его подходящим для этой цели?
4) Какой формат данных из рассматриваемых лучше всего подходит для хранения и передачи простых таблиц без вложенных структур, и почему?
5) Какие ограничения существуют при использовании JSON для представления данных, и как они могут влиять на обработку информации?

Лабораторная работа №3

Работа в Tkinter

Введение

В данной работе будет рассмотрено проектирование пользовательского интерфейса. Хотя почти все работы из этого курса выполняются в командной строке (он же Terminal), проектирование интерфейса является важной задачей, особенно для программ, где важна наглядность ввода данных. Все более-менее крупные языки программирования так или иначе имеют такой функционал, позволяющий делать ваши приложения более дружелюбными для пользователя.

В Python для проектирования интерфейса есть несколько решений, но здесь мы рассмотрим Tkinter. Tkinter – библиотека с долгой историей разработки, что тянется ещё из 80-х, она была адаптирована для множества языков. Для нас важными являются две вещи: Tkinter поставляется в составе дистрибутива Python, начиная с появления Python 3.x, и его не надо дополнительно скачивать и устанавливать, а также его богатый функционал, которого должно хватить для абсолютного большинства ваших задач.

Полезные ссылки:

- Официальная документация по библиотеке tkinter: https://docs.python.org/3/library/tk.html
- Официальная документация по библиотеке random: https://docs.python.org/3/library/random.html
- Об использовании tkinter: https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter

Описание лабораторной работы

Цель работы

Получить практические навыки о программировании пользовательского интерфейса на примере библиотеки tkinter, а также о генерации численно-буквенных последовательностей с использованием библиотеки random.

Указания к выполнению работы

Работа состоит из двух частей. Первая часть отведена под проектирование внешнего вида пользовательского интерфейса программы с размещением необходимых элементов. Вторая часть заключается в реализации полезной функции программы. Имеется дополнительное задание по расширению функционала написанной программы, которое выполняется по желанию.

Оформление отчета

Отчёт должен быть составлен в соответствии с требованиями, представленными во введении.

Ход работы

Код простой программы, которая выводит фрейм с текстом:

```
import tkinter as tk


window = tk.Tk()
window.title("My title")
window.geometry("320x240")

label = tk.Label(text="Hello ITMO")
label.pack()

window.mainloop()
```

Появляется созданный фрейм, имеющий заголовок, указанные размеры и текст в метке. Обратите внимание, что оформление элементов интерфейса зависит от вашей операционной системы. Например, в MacOS это окно будет выглядеть соответственно интерфейсу этой ОС от Apple.

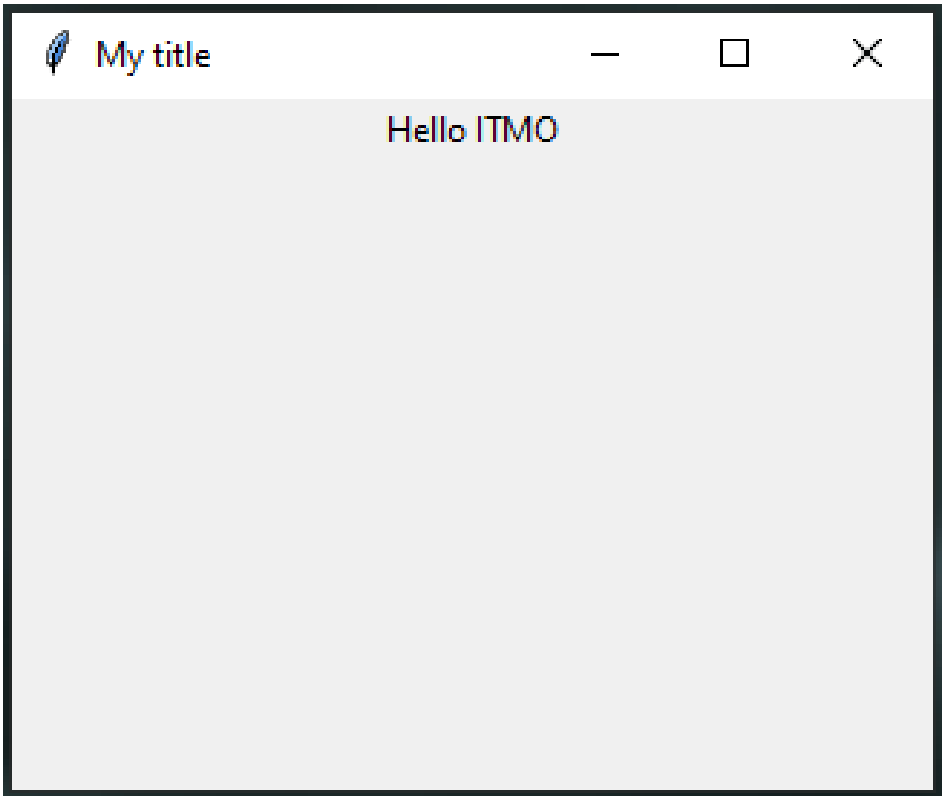


Рисунок 3.1 – Изображение пустого фрейма

Следующая после import строка создаёт объект класса `Tk` и присваивает его переменной `window` – это тот самый фрейм, который будет выводиться при запуске. Теперь можно применять методы, описанные в библиотеке Tkinter, к объекту window. Вторая и третья строки демонстрируют одни из базовых методов:

`title()` устанавливает заголовок окна, а `geometry()` задаёт размер окна в пикселях.

Далее, `label` – объект класса `Label` с одним аргументом text. `label.pack()` – это метод, который отвечает за автоматическое размещение элемента. Элементы можно размещать и вручную, задавая положение по координатам или по соотношению. В целом, заполнение фрейма элементами выполняется по схеме «*создать элемент – разместить элемент*».

`Label` имеет много параметров, которые можно указывать в скобках. Это справедливо и для многих других типов элементов интерфейса. Вот пример ещё пары параметров:

```
label = tk.Label(
    text="Hello ITMO",
    background="#FFFF00",
    font=("Verdana", 16)
)
```

Вид метки изменился: у неё появился фон и шрифт Verdana.

Рисунок 3.2 – Изменения в оформлении надписи

Программу стоит немного модифицировать, чтобы она могла делать что-нибудь полезное. К примеру, выводила информацию о том, сколько пользователю лет. Необходимо добавить ещё два элемента: поле для ввода данных и кнопку,

а уже добавленную метку надо немного изменить. Информацию выведем во всплывающем окне, хотя это можно делать и в тексте метки.

Над последней строкой будет размещён следующий фрагмент кода:

```
entry = tk.Entry(window, width=10)
entry.insert(0, "1990")
entry.pack()

btn_calc = tk.Button(window, text="Узнать", com-mand=calc)
btn_calc.pack()
```

Здесь под переменными `entry` и `btn_calc` скрываются поле ввода текста и кнопка, которая будет инициировать появление окна с результатом. Поле ввода теста – объект класса `Entry`, здесь задается его длина в символах. Помимо этого, можно указать значение по умолчанию через метод `insert()`, где `0` – номер строки, а «`1990`» - значение поля по умолчанию.

С кнопкой ситуация примерно такая же, как и с `Label` – у объекта класса Button схожий набор параметров, но есть один уникальный, `command`. Он отвечает за то, какой код будет исполняться по нажатию кнопки. Удобнее всего обернуть его в функцию, имя которой и указать как значение аргумента `command`.

Сама функция `calc()` должна брать значение из поля `entry`, брать текущий год при помощи библиотеки `datetime`, а затем выводить во всплывающем окне. Код данной функции будет следующий:

```
def calc():
    year = int(entry.get())

    current_year = int(
        datetime.date.today().isoformat()[:4]
    )

    tk.messagebox.showinfo(
        "Результат",
        f"Вам {current_year - year} полных лет. "
    )
```

В первой строке значение берется из поля ввода при помощи метода `get()`. Благодаря тому, что было указано значение по умолчанию, даже если пользователь ничего не введет, то год всё равно будет получен. Текущий год рассчитывается из метода, который выводит текущую дату, необходимо отбросить всё, кроме года, в данном случае, взять первые 4 символа. Обе переменные необходимо привести к типу int, поскольку по умолчанию они имеют строковый тип.

Для всплывающих сообщений у `tkinter` есть подмодуль `messagebox` – это необходимо импортировать отдельно. Он позволяет выводить информационные сообщения, такие как предупреждения или ошибки – при помощи методов

`showwarning()` и `showerror()`. Но возраст – это просто информация, поэтому выведем окошко информации через `showinfo()`.

Наконец, необходимо рассмотреть возможность добавления фонового изображения. Фон – это тоже метка, в которую будет подгружен файл с изображением. Формат JPG здесь читаться ***не будет***, следует использовать PNG!

```
bg_img = tk.PhotoImage(file="gradient.png")

lbl_bg = tk.Label(window, image=bg_img)
lbl_bg.place(x=0, y=0, relwidth=1, relheight=1)
```

Изображение здесь – объект класса `PhotoImage`, в аргументы которому прописывается относительный путь до используемого изображения. Далее создаётся ещё один объект `Label`, которому мы должны передать переменную с изображением, а также разместить. Объекты в `tkinter` размещаются послойно, поэтому этот `Label` надо определить до всех остальных элементов главного окна, после чего разместить через метод `place()` – при данных значениях параметров изображение будет размещено относительно центра окна.

Однако разрешение у окна и у изображения совпадает, поэтому это будет заметно, только если попытаться изменить размер окна курсором за рамку, как с \ ЫЛюбым другим окном в вашей операционной системе. Масштабирование фона сделать можно, но это потребует подключения дополнительной библиотеки `PIL`, поэтому здесь не рассматривается. Итого, весь код программы будет следующим:

```
import tkinter as tk
from tkinter import messagebox
import datetime

def calc():
    year = int(entry.get())
    current_year = int(
        datetime.date.today().isoformat()[:4]
)

    tk.messagebox.showinfo(
        "Результат",
        f"Вам {current_year - year} полных лет. "
    )
window = tk.Tk()
window.title("My title")
window.geometry("320x240")
bg_img = tk.PhotoImage(file="gradient.png")
```

```
lbl_bg = tk.Label(window, image=bg_img)
lbl_bg.place(x=0, y=0, relwidth=1, relheight=1)

label = tk.Label(text="Ваш год рождения: ", back-
ground="#00FF00", font=("Verdana", 16))
label.pack()

entry = tk.Entry(window, width=10)
entry.insert(0, "1990")
entry.pack()

btn_calc = tk.Button(window, text="Узнать", com-mand=calc)
btn_calc.pack()
window.mainloop()
```

А результат выполнения – таким:

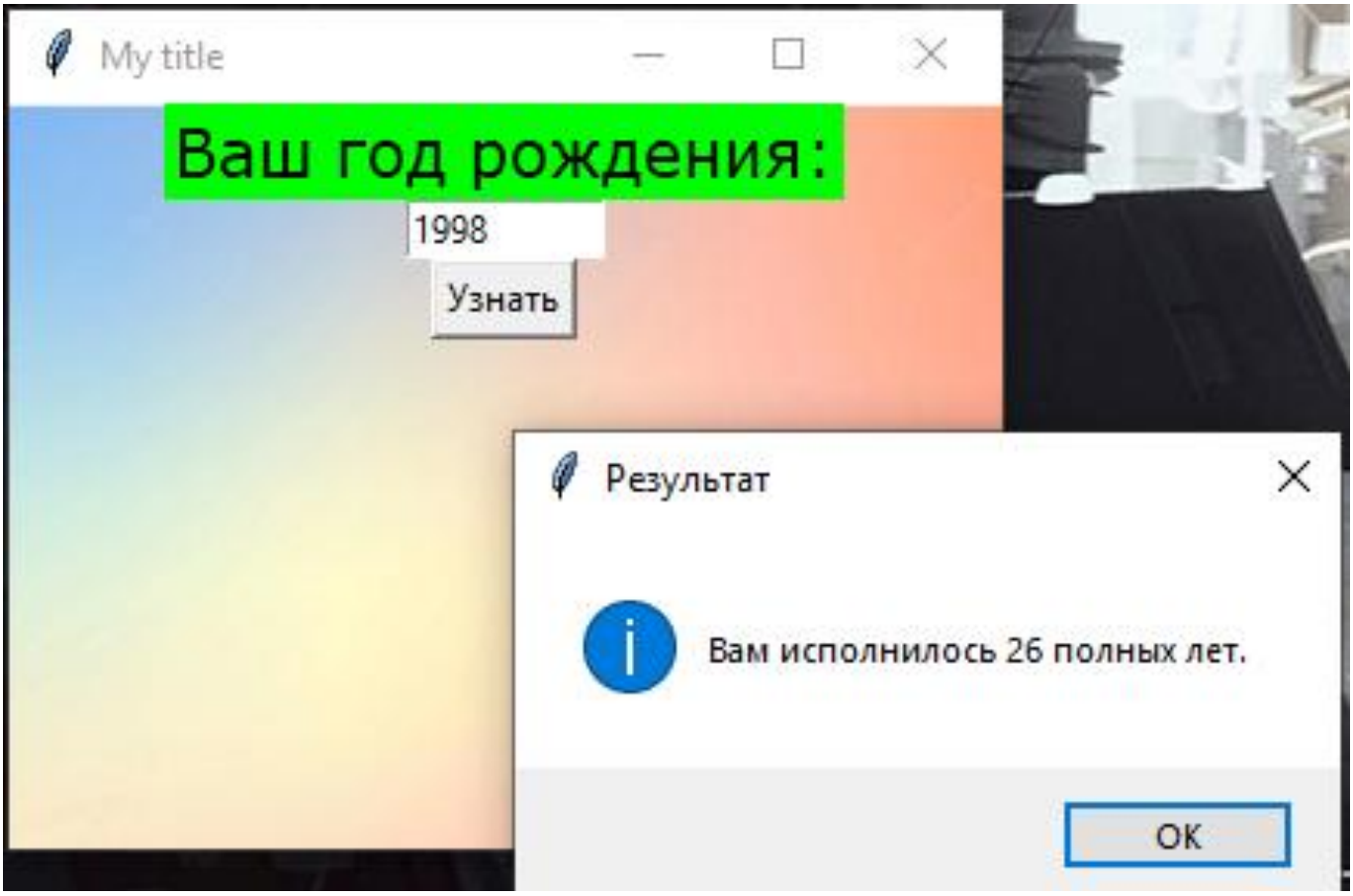


Рисунок 3.3 – Результат выполнения программы с реализованной полезной функцией

Другие функции и элементы, которые можно создать при помощи tkinter, указаны в официальной документации к библиотеке.

Задание и варианты

Используя `tkinter`, реализовать `keygen` – программу, генерирующую ключ для некоего абстрактного программного обеспечения (мы осуждаем пиратство) с соответствующим оформлением пользовательского интерфейса.

- Выберите любимую игру, найдите по ней арт или связанную картинку в поисковике.
- Реализуйте форму генератора ключа, которая должна включать в себя, как минимум, поле для генерируемого ключа, кнопку запуска, найденную картинку на фоне и поле ввода текста, если того требует вариант.
- Реализуйте генератор ключа. Ключ состоит из набора символов, состоящих из латинских букв **A-Z** и цифр **0-9**. В зависимости от варианта может потребоваться ввод первой части ключа (указан в варианте). Ключ генерируется по некоторым правилам. В заданиях со сдвигом считать, что буквы и цифры последовательно как бы нанесены на бесконечную ленту, которую можно двигать влево и вправо. DEC – число в десятичном формате, HEX – число в шестнадцатеричном формате.

Таблица 3.1 – Варианты задания лабораторной работы № 3

Вар.	**Вводимая часть**	**Формат ключа**	**Условие**
1	нет	XXXXX-XXXXX-XXXXX	Каждый блок имеет две цифры и три буквы в случайном порядке *Пример:* FX26N-N3RT7-AZ0J8
2	нет	XXXX-XXXX-XXXX-XXXX	Каждый блок имеет одну цифру и три буквы в случайном порядке *Пример:* AB8U-Z0MI-7FYK-K9GT
3	HEX-число, 5 знаков	XXXXX-XXXXX-XXXXX XX	Вводная часть ключа - число в HEX, которое необходимо перевести в DEC[1]. Первые три цифры числа в DEC должны быть по одной в каждом блоке, последние две цифры – в конце ключа *Пример*: 54CD1(HEX)=347345(DEC) DS**3**BG-**4**09KJ-T6**7**K8 **45**
4	нет	XXXX-XXXX-XXXX	Назначить весовые коэффициенты символам, генерировать с учётом, чтобы сумма весовых коэффициентов одного блока попала в интервал[2] *Пример*: пусть A=1, B=2,… Пусть интервал – 30…35 YABD-NBCO-DGIK Суммы: 32-34-31
5	Первый	XXXXX-XXXXX-	Опираясь на введённый фрагмент ключа,

[1] Можно воспользоваться конструкцией `dec = int(hex, 16)`.

[2] Весовые коэффициенты, а также интервалы указать самостоятельно.

	блок ключа	XXXXX	сгенерировать остаток ключа так: 2 блок – сдвиг на 3 символа вправо, 3 блок – сдвиг на 5 символов влево JINOS-MLQRV-EDIJN
6	нет	XXXXX-XXXX-XXXX	Назначить весовые коэффициенты символам, генерировать с учётом, чтобы среднее значение букв из одного блока попало в интервал *Пример*: пусть A=1, B=2,… Пусть интервал – 10…15 YAND-NZCQ-WGIK Средние: 11-15-12
7	Слово, 6 букв	XXX-XXXXXX-XXX	1 и 3 блок – только буквы, взятые из введённого слова, 2 блок - только цифры, соответствующие порядковым номерам букв в алфавите (десятки отбросить) *Пример*: ввод «MASTER» TMR 319058 AES
8	DEC-число, 3 знака	XXXXX-XXXX-XXX-XX	1 блок - случайная комбинация букв, каждый последующий блок убирает последнюю букву и выполняет сдвиг на последующую цифру из введённого числа. Направление сдвига чередуется. *Пример*: ввод «123» DRITF-ESJY-CPR-FS (сдвиг 1 вправо-2 влево-3 вправо)
9	нет	XX XXXXXXX XX	1 и 3 блок - порядковые числа двух букв из алфавита, выбранных случайно (не больше 26) - обозначение границ интервала, 2 блок заполняется буквами из этого интервала в случайном порядке *Пример*: выпали A и J 01 DEBBHCI 10
10	DEC-число, 6 знаков	XXXXX-XXXXX XXXX	1 и 2 блок должны содержать 4,5,6 и 1,2,3 цифры введенного числа соответственно, остальное - случайные буквы, 3 блок - результат сложения чисел, получившихся в 1 и 2 блоках *Пример*: ввод «726911» 276DL-191GO-0467

Дополнительное задание

Добавить музыку (8-bit желательно) на фоне и анимацию.

Примечания

Для генерации рекомендуется использовать библиотеку `random`. Полезные функции:

- `randint()` – генерация числа из диапазона.
- `choice()` – случайный элемент списка.
- `shuffle()` – перемешать список.
- `randrange()` – случайное число из набора чисел.

Контрольные вопросы

1) Какие вы знаете методы для позиционирования созданных элементов интерфейса?
2) Для чего нужно проектировать пользовательский интерфейс?
3) Каким методом задаётся размер фрейма?
4) Какой аргумент объекта `Button` служит для передачи в него функции с исполняющим кодом?

Лабораторная работа № 4

Задача о рюкзаке

Введение

Задача о рюкзаке является классическим примером комбинаторной задачи, в которой, как очевидно из названия, целью является размещение в условном «рюкзаке» как можно больше предметов. Будет рассмотрен частный упрощённый случай такой задачи с целочисленными значениями параметров, но сперва стоит разобраться с обобщённой постановкой задачи.

Полезные ссылки:

- Описание задачи о рюкзаке: https://proglib.io/p/python-i-dinamicheskoe-programmirovanie-na-primere-zadachi-o-ryukzake-2020-02-04
- Метод ветвей и границ: https//habr.com/ru/articles/560468/

Описание лабораторной работы

Цель работы

Получить практические навыки в реализации алгоритмов по их формальному и схематическому описанию.

Указания к выполнению работы

Работа состоит из одной части, в которой предлагается реализовать в виде программы один из 4 предложенных алгоритмов. Присутствует дополнительное задание, которое предлагает решить ту же задачу с более жёсткими ограничениями, выполняемое по желанию.

Оформление отчета

Отчёт должен быть составлен в соответствии с требованиями, представленными во введении.

Ход работы

Даны следующие положения:

1) Набор из n объектов o_n, имеющих ценность $v_i > 0$ и вес $w_i > 0$.
2) Условное хранилище, обладающее ограниченной вместимостью $W > 0$.
3) Объект нельзя поместить в хранилище частично - либо целиком, либо не помещать вовсе.
4) Каждый объект можно взять только один раз, после чего он пропадает из набора и не участвует в дальнейших выборах.

Найти: подмножество объектов o_k, где $k < n$, такое, что $\sum_{i=0}^{k} w_i \leq W$ при максимизации значения $\sum_{i=0}^{k} v_i$. Возможные варианты решения данной задачи:

Полный перебор объектов

При полном переборе выполняется подбор различных комбинаций, последовательно или с генерацией случайных последовательностей. Подбор осуществляется по следующей схеме: выбрав объект o_i, сперва значение W уменьшаем на w_i, после чего, если $W \geq 0$, мы помещаем объект в хранилище, убирая его из набора. Если значение $W < 0$, то подбор прекращается, после чего результат можно представить как решение. Очевидно, что первый же удовлетворительный результат вряд ли будет оптимальным, для поиска наилучшего решения надо перебрать все $C = n!/((n-k)! \cdot k!)$ комбинаций, что, несомненно, займет некоторое время, которое будет расти в геометрической прогрессии с увеличением набора n.

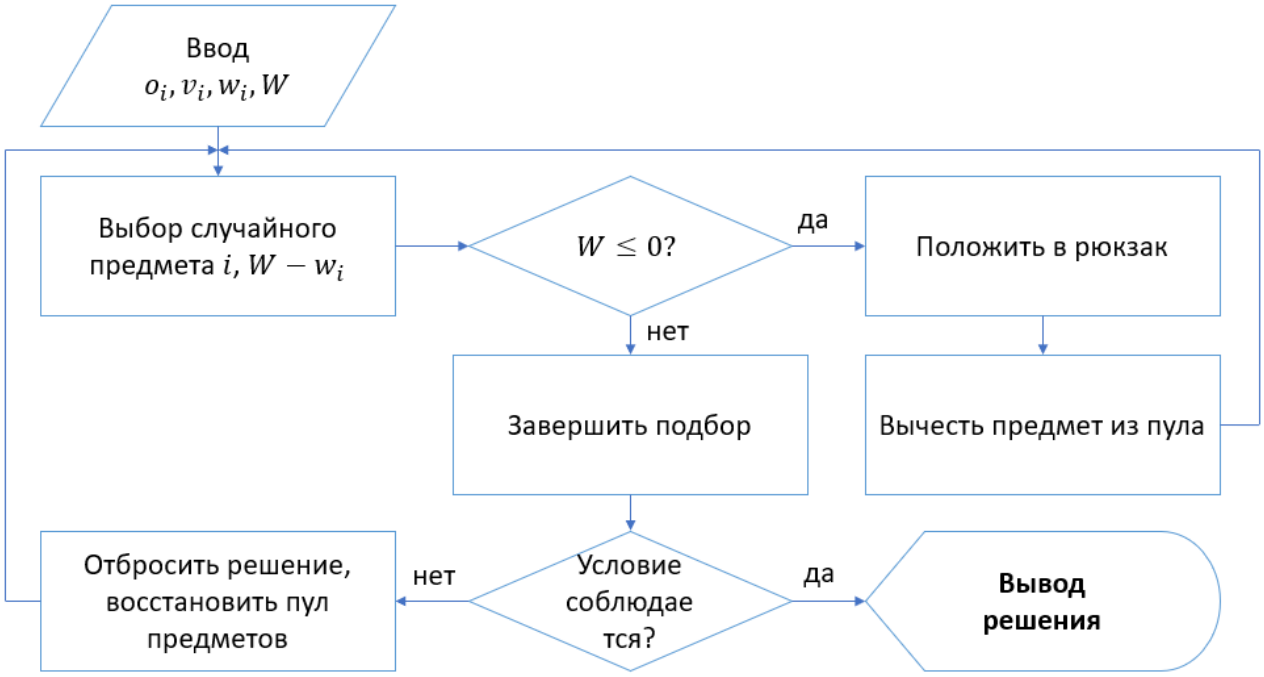


Рисунок 4.1 – Блок-схема алгоритма полного перебора

Метод ветвей и границ

Метод ветвей и границ – вариация метода перебора, основное отличие которой заключается в отбрасывании заведомо неверных гипотез. Здесь выполняется построение дерева решений, которое анализируется в ширину на каждом ярусе его ветвей. Так, первый ярус дерева будет содержать n гипотез, второй ярус дерева – уже $n(n-1)$, третий – $n(n-1)(n-2)$ и т. д. Но для каждого узла дерева выполняется расчёт стоимости и суммарного веса и, если узел не удовлетворяет этой оценке, то данную ветку можно не рассматривать. Таким образом, на каждом этапе количество узлов будет сокращаться, и, в конце концов, будет предложено одно или несколько наиболее оптимальных решений. Этот метод лучше всего применим в случаях, когда стоимости объектов очень сильно разнятся, в противном же случае экономия времени по сравнению с полным перебором может быть несущественной.

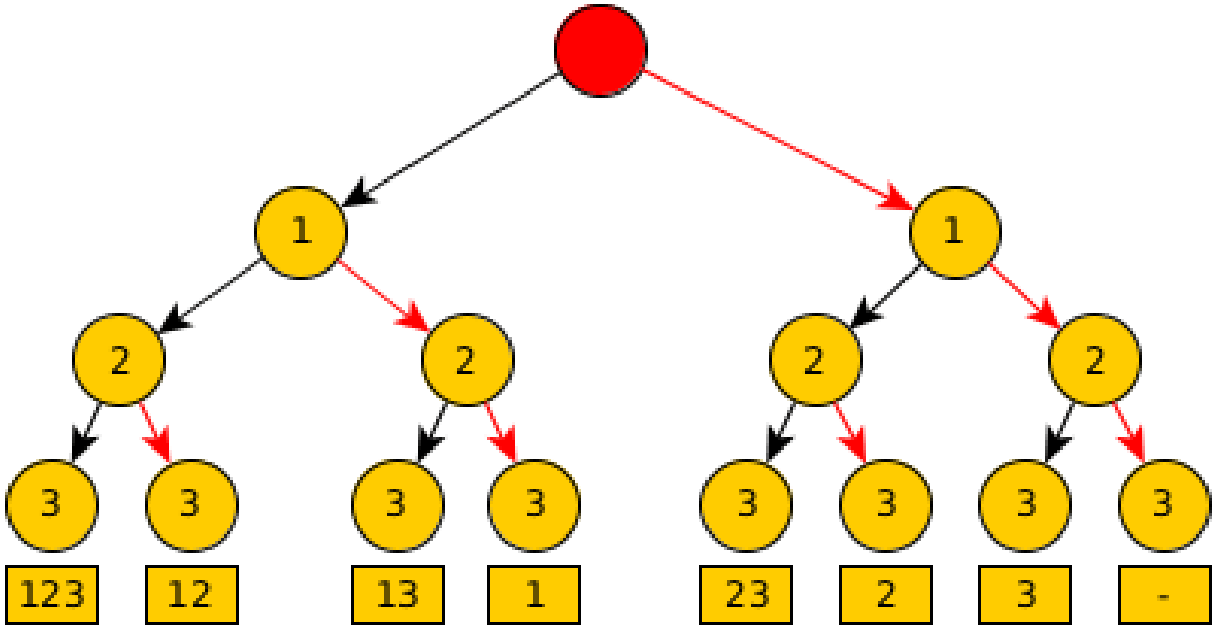


Рисунок 4.2 – пример древовидного представления решений

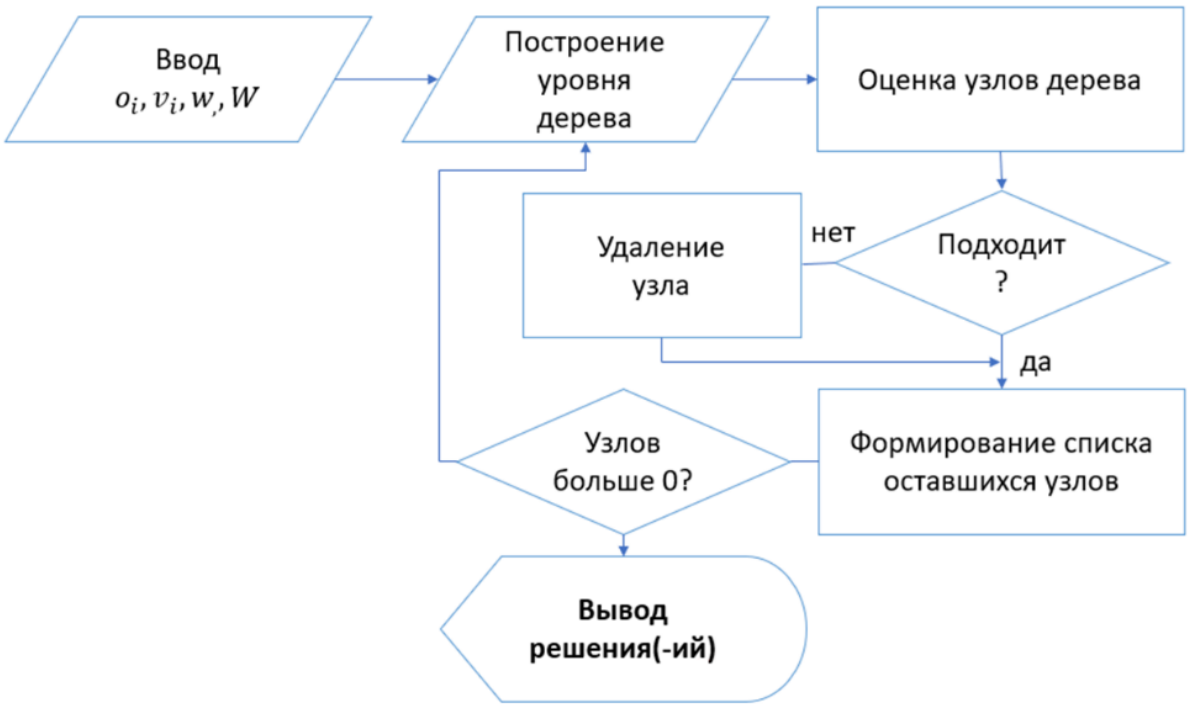


Рисунок 4.3 – Блок-схема метода ветвей и границ

«Жадный» алгоритм

Суть этого алгоритма – подбирать объекты, исходя из их максимальной ценности. Объекты ранжируются по их ценности, либо, что предпочтительнее, удельной ценности $w'_i = v_i/w_i$. После чего начинаем помещать объекты в хра-

нилище, начиная с самого ценного, пока хранилище не заполнится. Этот алгоритм прост в реализации, но не гарантирует оптимального решения.

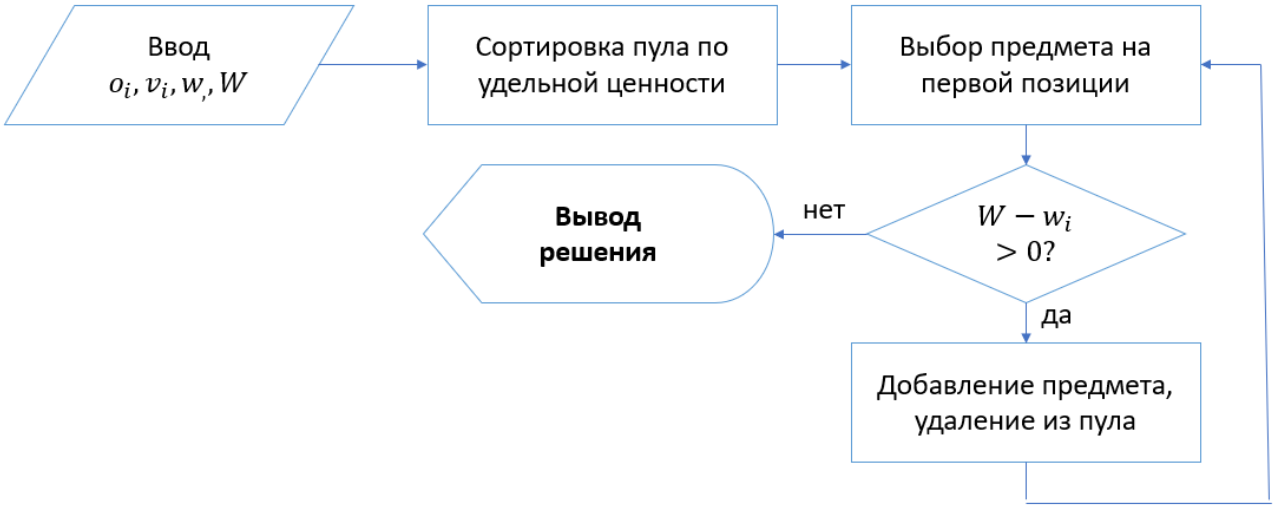


Рисунок 4.4 – Блок-схема жадного алгоритма

Динамическое программирование

В данном алгоритме последовательно рассматривается набор хранилищ от 1 до W с последовательным увеличением набора предметов. Анализ начинается с объекта o_1 и выполняется проверка на входимость этого объекта в хранилища вместимостью 1, 2, …, W с расчётом ценности для каждого случая. Затем добавляется ещё один объект, набор увеличивается до (o_1, o_2), который также проверяется на входимость в хранилища от 1 до W. Данная процедура повторяется до тех пор, пока рассматриваемый набор объектов не будет равен исходному. В итоге должна получиться таблица в n строк и W столбцов, нижняя правая ячейка которой будет содержать искомое решение. Например, для набора объектов, представленного в таблице 4.1 и хранилища вместимостью W=4 данная таблица будет иметь следующий вид (таблица 4.2).

Таблица 4.1 – Таблица весов и стоимости объектов

Объект	**Вес w_i**	**Стоимость v_i**
Шкаф (Ш)	3	700
Табурет (Т)	1	300
Стол (С)	2	500
Комод (К)	2	550

Таблица 4.2 – Таблица с результатом расчётов

Предмет	1	2	3	4
Шкаф / 3 / 700	0	0	700 (Ш)	700 (Ш)
Табурет / 1 / 300	300 (Т)	300 (Т)	700 (Ш)	1000 (Ш+Т)
Стол / 2 / 500	300 (Т)	500 (С)	800 (Т+С)	1000 (Ш+Т)
Комод / 2 / 550	300 (Т)	550 (К)	850 (Т+К)	**1050 (С+К)**

Проанализировав данный алгоритм, можно заключить, что значение суммы для каждой ячейки этой таблицы будет:

$$T(i,j) = \max_{v}\begin{cases} T(i-1,j) \\ v_0 + T(i-1, j-w_0)' \end{cases}$$

где v_0 и w_0 – ценность и вес текущего (последнего добавленного в набор) предмета. Есть и другие алгоритмы для решения этой задачи, но они в данной работе не рассматриваются ввиду сложности их реализации.

Задание и варианты

Дан набор предметов, представленных в таблице 4.3.

Таблица 4.3 – Стартовый набор предметов

Предмет	Обозначение	Размер в ячейках	Очки выживания
Винтовка (rifle)	r	3	25
Пистолет (pistol)	p	2	15
Боекомплект (ammo)	a	2	15
Аптечка (medkit)	m	2	20
Ингалятор (inhaler)	i	1	5*
Нож (knife)	k	1	15
Топор (axe)	x	3	20
Оберег (talisman)	t	1	25
Фляжка (flask)	f	1	15
Антидот (antidot)	d	1	10*
Еда (supplies)	s	2	20
Арбалет (crossbow)	c	2	20

Для определения оптимальности решения в алгоритмах перебора вводится условие: каждый добавленный предмет увеличивает очки выживания (ценность) на свою w_i, но каждый оставленный – их отнимает на те же w_i. Итоговое количество очков выживания должно быть положительным с учётом стартовых очков (см. варианты).

Также, в зависимости от варианта, в наборе может быть требование присутствия антидота (d) или ингалятора (i). В таком случае предмет убирается из стартового набора, хранилище уменьшается на 1, а количество очков выживания увеличивается на 10 или 5 соответственно.

Реализовать один из предложенных алгоритмов и представить решение в виде набора ячеек, например:

```
[r], [r], [r]
[c], [c], [t]
[m], [m], [k]

Итоговые очки выживания: 5
```

Каждый набор квадратных скобок обозначает 1 ячейку, поэтому винтовка (r), занимающая 3 ячейки, повторяется 3 раза, арбалет (c) – 2 раза и т. д. Варианты задания представлены в таблице 4.4.

Таблица 4.4 – Варианты задания для лабораторной работы № 4

Вариант	Хранилище	Требование	Очки выживания
1	2×4	-	15
2	3×3	ингалятор	10
3	2×4	антидот	10
4	3×3	-	15
5	2×4	ингалятор	20
6	3×3	антидот	15
7	2×4	-	15
8	3×3	ингалятор	15
9	2×4	антидот	20
10	3×3	-	10

Дополнительное задание

Найти решение или доказать его отсутствие для случая с инвентарём в 7 ячеек.

Контрольные вопросы

1) Какой алгоритм, на ваш взгляд, наиболее прост в реализации на языке программирования?

2) Что такое алгоритм?
3) Усложнит или упростит решение, на ваш взгляд, тот факт, что мы можем брать предмет неограниченное количество раз?

Лабораторная работа №5

Работа с регулярными выражениями

Введение

Регулярные выражения – это формальный язык, используемый для манипуляций с текстовыми данными. Этот язык основан на применении шаблонов и метасимволов и при правильном использовании является мощным инструментом поиска информации. Этот язык не характерен исключительно для языка Python, данный стандарт формализованных конструкций применяется практически во всех распространённых языках программирования или средах обработки данных. Наиболее частая задача, решаемая регулярными выражениями – парсинг, к примеру, отданной с сервера страницы с базой данных.

Полезные ресурсы:

- Статья о регулярных выражениях: https://habr.com/ru/articles/349860/
- Сайт для отработки регулярных выражений: https://regex101.com

Описание лабораторной работы

Цель работы

Получить практические навыки использования регулярных выражения для поиска необходимой информации в файлах с различным содержимым.

Указания к выполнению работы

Работа состоит из трех частей. Первая часть отведена под поиск слов или цифровых выражений в простых тестовых данных. Вторая часть заключается в поиске конструкций в HTML-коде страницы. Третья часть посвящена формированию из сырых данных структурированной таблицы в виде файла формата CSV. Имеется дополнительное задание по поиску полезных данных в большом файле, заполненном случайными символами, которое выполняется по желанию.

Оформление отчета

Отчёт должен быть составлен в соответствии с требованиями, представленными во введении.

Ход работы

В Python за использование регулярных выражений отвечает библиотека `re`. Она не требует скачивания, поскольку поставляется вместе с файлами интерпретатора Python.

Основные используемые методы, реализованные в библиотеке re:

1) `match(<шаблон>, <строка>)` – поиск шаблона по началу строки
2) `search(<шаблон>, <строка>)` – поиск шаблона по всей строке

3) `findall(<шаблон>, <строка>)` – выводит все вхождения шаблона в строке в виде списка
4) `split(<шаблон>, <строка>)` – разбивает строку на список подстрок по шаблону. Существует параметр maxsplit=<число> - разбиемние строки на указанное число раз с игнорированием оставшейся части строки.
5) `sub(<шаблон1>, <шаблон2>, <строка>)` – замена в строке одного шаблона на другой.
6) `fullmatch(<шаблон>, <строка>)` – проверка на соответствие шаблона исходной строке.

Применение этих методов для простых шаблонов не вызовет никакой сложности. Пусть есть следующий код:

```
import re

line = 'давайте будем анализировать строки 24 часа в сутки'
result = re.search(r"будем", line)
print(result)
```

Здесь дана строка, содержащая фразу. К ней применяется метод `search()` с шаблоном «будем». Буква `r` перед шаблоном – это флаг, отключающий экранирование символов. Вывод `result` будет следующим:

```
<re.Match object; span=(8, 13), match='будем'>
```

Метод нашёл вхождение шаблона «будем» в строке на позиции в 8 по 12 символ включительно. Эти числа можно получить, если применить к result метод `span()`:

```
print(result.span())
```

Результат:

```
(8, 13)
```

Пример другого метода, `findall()`:

```
line = 'давайте будем анализировать строки 24 часа в сутки'
result = re.findall(r"ва", line)
print(result)
```

Даст результат:

```
['ва', 'ва']
```

Данный метод вернёт все вхождения в виде списка.

На данном этапе поведение методов практически не отличается от стандартных методов, применяемых к строкам. Однако, когда заходит речь о более обобщённых шаблонах, то разница становится очевидна. Во-первых, можно использовать диапазоны, которые записываются следующим образом:

`[уэл]` – вхождения букв «у», «э» или «л»

`[а-о]` – вхождения букв в диапазоне от «а» до «о»

`[0-9]` – вхождения цифр в диапазоне от 0 до 9

`[^абв]` – вхождения букв, не являющихся «а», «б» или «в»

Примечание: буква «ё» находится вне диапазона кириллицы, поэтому для её включения необходимо указывать интервал `[а-яё]`.

Пример – в рассматриваемой строке надо подсчитать количество букв «а» и «т» суммарно:

```
result = re.findall(r"[ат]", line)
print(len(result))
```

Результат: `11`. Наиболее часто используемые интервалы в итоге стали спецсимволами:

`\d` – любая цифра, [0-9]

`\s` – пробелы и символы табуляции

`\w` – буквы, цифры и знак «_», [а-яА-Я0-9_]

Версии этих спецсимволов с заглавной буквой делают результат прямо противоположным: `\D` – любая НЕ цифра, `\S` – любой НЕ пробел и т.д.

Во-вторых, квантификаторы, используемые для удобства записи выражения. Есть 3 основных односимвольных квантификатора:

`?` – 0 или 1 вхождение

`+` - 1 или более вхождений

`*` - 0 или более вхождений

К примеру, метод `findall(r"\d", line)` вернёт список из двух элементов: `['2', '4']`, а `findall(r"\d+", line)` – уже из одного `['24']`. В первом случае происходит поиск по одному символу, а во втором уже по группе цифр. Количество вхождений можно задавать и более точно:

`{3}` – ровно 3 вхождения

`{1,6}` – от 1 до 6 вхождений

`{,5}` – не более 5 вхождений

`{2,}` – не менее 2 вхождений

Код `re.findall(r"\w{5}", line)` выведет все пятибуквенные последовательности из данной строки:

```
['давай', 'будем', 'анали', 'зиров', 'строк', 'сутки']
```

Если стоит задача найти именно пятибуквенные слова, то шаблон надо модифицировать ещё одним спецсимволом - `\b`. Он выполняет поиск по началу или по концу слова.

```
result = re.findall(r"\b\w{5}\b", line)
['будем', 'сутки']
```

Применение данных правил может быть проиллюстрировано на примере. Пусть есть следующие данные:

```
line = '''Renault Logan, 2015 г.в., в605ен178
        Ford Fusion, 2017 г.в., а 301 рр 198
        Volkswagen Polo, 2013 г.в., у617оо-147'''
```

Необходимо вытащить все регистрационные номера, но все они написаны в разном формате. В первом случае выражение будет простым:

```
result = re.findall(r"\w\d{3}\w{2}\d{3}", line)
```

Однако в следующей строке группы символов уже разделены пробелами, поэтому выражение нужно модифицировать:

```
result = re.findall(r"\w ?\d{3} ?\w{2} ?\d{3}", line)
```

Были добавлены пробелы с квантификатором «?» — это означает, что в этом месте может быть, а может не быть пробела. Данное выражение выведет первый и второй номерной знак, но есть ещё и третий, где последний блок отделён дефисом. Ещё одна модификация выражения:

```
result = re.findall(r"\w ?\d{3} ?\w{2}[ -]?\d{3}", line)
```

Вместо последнего пробела вводится набор из пробела и дефиса, таким образом, в выражении в этом месте может стоять пробел, дефис, или не быть ничего. Все три номера выводятся:

```
['в605ен178', 'а 301 рр 198', 'у617оо-147']
```

Задача усложняется: к имеющимся строкам прибавляется ещё одна:

```
line = '''Renault Logan, 2015 г.в., в605ен178
        Ford Fusion, 2017 г.в., а 301 рр 198
        Volkswagen Polo, 2013 г.в., у617оо-147
        Lada Granta, 2011 г.в., с313ур 98'''
```

В данном случае на конце уже две цифры, и в выражение снова необходимо вносить коррективы:

```
result = re.findall(r"\w ?\d{3} ?\w{2}[ -]?\d+", line)
```

Здесь произошла замена `\d{3}` на `\d{2,3}` – может быть 2 или 3 вхождения цифры. К счастью, по стандарту номерных знаков в этом месте может быть только 2 или3 цифры, значит, эту часть трогать больше не нужно. Задача усложняется дальше ещё одной строкой:

```
line = '''Renault Logan, 2015 г.в., в605ен178
        Ford Fusion, 2017 г.в., а 301 рр 198
        Volkswagen Polo, 2013 г.в., у617оо-147
        Lada Granta, 2011 г.в., с313ур 98
        Иж-21251, 1991 г.в., д 2139 ЛГ'''
```

Это номерной знак старого образца, и сейчас они встречаются довольно редко. Тем не менее, нужно подредактировать выражение и под него тоже. Этот номер состоит из одной буквы, 4 цифр и 2 букв.

```
result = re.findall(r"\w ?\d{3,4} ?\w{2}[ -]?(?:\d{2,3})?", line)
```

Была введена конструкция `(?:<…>)`, которая является группировкой элементов, ведь теперь последний блок из цифр может быть, а может и не быть. А кроме того, первый блок из цифр теперь может принимать как 3, так и 4 цифры. Все номера выводятся:

```
['в605ен178', 'а 301 рр 198', 'у617оо-147', 'с313ур 98', 'д 2139 ЛГ']
```

Так, постепенно, шаг за шагом модифицируя регулярное выражение, можно добиться довольно точного результата поиска.

Задание и варианты

1. Откройте файл `task1_ru.txt` или `task1_en.txt`. Используя регулярные выражения, выполните задание для своего варианта из списка ниже:

   Вариант 1. Все слова от 3 до 5 букв.
   Вариант 2. Все слова, начинающиеся с большой буквы.
   Вариант 3. Все слова, после которых стоит запятая.
   Вариант 4. Все слова, в которых есть дефис.
   Вариант 5. Все числа, целые и дробные.
   Вариант 6. Все целые числа больше 100.
   Вариант 7. Все слова, начинающиеся с буквы с (условно).
   Вариант 8. Все отрицательные числа, целые и дробные.
   Вариант 9. Все слова, заканчивающиеся на букву а (условно).
   Вариант 10. Все слова, после которых стоит какой-либо символ пунктуации.

2. Откройте файл `task2.html`. Используя регулярные выражения, выполните задание для своего варианта из списка ниже:

   Вариант 1. Все открывающие теги без повторений.
   Вариант 2. Все закрывающие теги без повторений.
   Вариант 3. Все ссылки.
   Вариант 4. Все ссылки в домене .com.
   Вариант 5. Все строки, находящиеся в атрибуте content любого тега.
   Вариант 6. Все ссылки на изображения.
   Вариант 7. Все заголовки статей.
   Вариант 8. Параметры используемых шрифтов: название, размер.
   Вариант 9. Все названия классов.
   Вариант 10. Количество пустых строк.

Откройте файл `task3.txt`. Здесь содержится таблица, состоящая из 5 полей: ID, фамилия, электронная почта, дата регистрации и сайт. Вот только пользователи вносили эти данные в абсолютно своём порядке, вследствие чего все данные перепутаны. Приведите эту базу данных в нормальный вид, расположив данные в вышеуказанном порядке. Для этого при помощи регулярных выражений выделите из файла данные одного типа, составьте из них таблицу и сохраните как файл формата `csv`.

Дополнительное задание

Есть файл со случайными символами – `task_add.txt`. Это бесполезные данные, белый шум, однако, в нём ещё можно найти что-то полезное. В этом файле скрыты 5 дат, 5 адресов электронной почты и 5 адресов сайтов. Каждый полезный элемент данных предваряется пробелом. Дата может быть в разных форматах и с разделителями вида – `/.`. Найдите все 15 фрагментов данных в файле, используя регулярные выражения.

Контрольные вопросы

1) Какой синтаксис используется для обозначения любого одного символа в регулярном выражении? Приведите пример использования этого выражения для поиска символов в строке.
2) Чем отличаются квантификаторы `*`, `+` и `?` в регулярных выражениях? Приведите примеры, чтобы объяснить различия между ними.
3) Как с помощью регулярного выражения найти все слова, начинающиеся с определенной буквы и оканчивающиеся на другую? Приведите пример шаблона и строки для демонстрации результата.
4) Как использовать группы (группировка) в регулярных выражениях и для чего они нужны? Приведите пример, где групповая выборка помогает получить полезную информацию из текста.

Лабораторная работа № 6

Объектно-ориентированное программирования

Введение

Объектно-ориентированное программирование (ООП) – это парадигма программирования, при которой программа строится на основе взаимодействующих объектов. В Python ООП является ключевой частью языка, предоставляющей гибкие возможности для моделирования и работы с данными. Основные принципы ООП включают инкапсуляцию, наследование, полиморфизм и абстракцию. Рассмотрим эти принципы более подробно.

Классы и объекты являются основными строительными блоками ООП в Python. Класс – это шаблон, описывающий общие черты и поведение, которыми обладают все его объекты. Он представляет собой чертеж для создания объектов, описывающий их свойства (атрибуты) и поведение (методы). Объект – это конкретный экземпляр класса, содержащий данные и методы, которые определены в классе. Создание класса позволяет задавать свойства и поведение, которые потом наследуются всеми его экземплярами.

Инкапсуляция – это принцип ООП, согласно которому данные и методы класса объединяются в единую сущность, ограничивая доступ к ним из внешнего кода. В Python инкапсуляция поддерживается через соглашение об именовании: атрибуты, начинающиеся с одного или двух символов подчеркивания (_ или __), считаются защищенными или приватными соответственно и не должны быть доступны извне. Атрибуты с двумя подчеркиваниями недоступны для прямого обращения, что помогает предотвратить случайное изменение данных и облегчает контроль над доступом к ним.

Наследование позволяет создать новый класс на основе существующего. Новый класс, называемый дочерним, наследует атрибуты и методы родительского класса, но может расширять или изменять их, добавляя свою функциональность. В Python наследование реализуется путем указания родительского класса в скобках после имени дочернего класса. Наследование позволяет создавать иерархии классов, а также использовать повторно и расширять код, уменьшая его дублирование. Python поддерживает как одиночное, так и множественное наследование. В случае множественного наследования класс может наследовать атрибуты и методы сразу нескольких родительских классов, что полезно для создания более гибких классов, но может также привести к сложности при решении конфликтов имен.

Полиморфизм – это способность методов работать с объектами разных типов. Он позволяет использовать один интерфейс для различных реализаций. Например, если у нескольких классов есть метод с одинаковым названием, то этот метод может вызываться для объектов любого из этих классов. Полиморфизм позволяет вызывать метод вне зависимости от типа объекта, облегчая добавление новых типов объектов в программу и упрощая обработку данных.

Абстракция – это принцип, согласно которому в классе скрываются детали реализации, предоставляя только нужный интерфейс. В Python абстракция реали-

зуется с помощью абстрактных базовых классов и методов, что позволяет определить интерфейс без конкретной реализации. Абстрактный метод указывает, что дочерние классы должны реализовать этот метод, а абстрактные классы нельзя создавать напрямую. Абстракция позволяет создавать шаблоны классов, которые определяют общий интерфейс для группы классов, задавая стандарты их поведения.

ООП в Python предоставляет мощный набор инструментов для структурирования и упрощения кода, особенно при работе со сложными данными и системами. Принципы инкапсуляции, наследования, полиморфизма и абстракции помогают организовать код, повысить его читаемость и облегчить сопровождение. В Python классы можно расширять, наследовать и изменять, что позволяет создавать гибкие и масштабируемые приложения.

Полезные ссылки:

- Официальная документация Python. Классы: https://docs.python.org/3/tutorial/classes.html
- Real Python. Основы ООП в Python: https://realpython.com/python3-object-oriented-programming/
- Python OOP на W3Schools: https://www.w3schools.com/python/python_classes.asp
- GeeksforGeeks. Полное руководство по ООП в Python: https://www.geeksforgeeks.org/python-oops-concepts/
- Курс по ООП в Python от Corey Schafer: https://www.youtube.com/watch?v=ZDa-Z5JzLYM

Описание лабораторной работы

Цель работы

Освоить основные принципы объектно-ориентированного программирования (ООП) в Python, включая создание классов и объектов, применение принципов инкапсуляции, наследования и полиморфизма, а также изучение особенностей множественного наследования и методов для управления доступом к данным внутри классов.

Указания к выполнению работы

Работа состоит из трех частей. В первой части необходимо создать класс с полями, в котором реализовать инициализатор и метод обработки данных. Во второй части следует спроектировать иерархию классов от изначально написанного класса, используя наследование. В третьей части нужно реализовать метод обработки данных в дочернем классе.

Оформление отчета

Отчёт должен быть составлен в соответствии с требованиями, представленными во введении.

Ход работы

Создадим класс «Файл». Класс включает в себя два статических поля класса, задающие имя класса (`class_name`) и счетчик экземпляров класса (`objects_count`). Данные поля не разделяются между экземплярами и относятся ко всему классу в целом. Также в классе определены три скрытых поля, определяющих имя файла (`_name`), размер в килобайтах (`_kbs`) и тип файла (`_ftype`). При инициализации экземпляра класса все эти поля должны быть заданы. Также при инициализации происходит увеличение на единицу счетчика экземпляров класса. В классе определены методы установления (`set_name`) и получения (`get_name`) имени файла, задания объема в килобайтах (`set_kbs`) и получения объема в килобайтах (`get_kbs`); получения типа файла (`get_ftype`) и информации о файле (`get_info`). Также в классе реализован вспомогательный метод преобразования размера файла из килобайтов в байты (`kbs2bytes`).

```
class File:
    class_name = 'File'
    objects_count = 0

    def __init__(self, name, kbs, ftype):
        self._name = name
        self._kbs = kbs
        self._ftype = ftype
        File.objects_count += 1

    def get_name(self):
        return self._name

    def set_name(self, n):
        self._name = n

    def get_kbs(self):
        return self._kbs

    def set_kbs(self, kbs):
        if kbs > 0:
            self._kbs = kbs
        else:
            self._kbs = 0.1

    def get_ftype(self):
        return self._ftype

    def get_info(self):
        print(self._name)
```

```
        print(f"Размер: {self._kbs} кБ")
        print(f'Формат: {self._ftype}')

    def kbs2bytes(self):
        print(f'Размер в байтах: {self._kbs * 1024}')
```

Создадим дочерний класс «Изображение», наследующий поведение класса «Файл». Добавим в класс два открытых поля, задающих высоту (**height**) и ширину (**width**) файла. Эти поля должны быть заданы при инициализации экземпляра класса, как и поля родительского класса. Также добавим методы задания ширины (**set_width**) и высоты (**set_height**) изображения и получения площади изображения (**get_amount**). Изменим поведение родительского метода получения информации о изображении (**get_info**), выполнив так называемую «перегрузку», то есть расширение возможностей метода. Также перегрузим системный метод равенства (**__eq__**), который вызывается, когда выполняется сравнения экземпляров класса с помощью операции «строгое равенство» (**==**).

```
class Image(File):
    class_name = 'Image'

    def __init__(self, name, kbs, ftype, height, width):
        super().__init__(name, kbs, ftype)
        self.height = height
        self.width = width

    def set_height(self, height):
        if height > 0:
            self.height = height
        else:
            self.height = 1

    def set_width(self, width):
        if width > 0:
            self.width = width
        else:
            self.width = 1

    def get_info(self):
        super().get_info()
        print(f'Тип: {Image.class_name}')
        print(f"Высота (пкс): {self.height}")
        print(f'Ширина (пкс): {self.width}')

    def get_amount(self):
```

```
        print(f'Площадь в пикселях: {self.height * self.width}')

    def __eq__(self, other):
        return self.height == other.height and self.width == other.width
```

Создадим экземпляр класса «Файл».

```
fl = File(f'Объект класса {File.class_name}', 11, 'TXT')
```

Вызовем методы созданного экземпляра.

```
fl.get_info()
fl.kbs2bytes()
```

Создадим два экземпляра класса «Изображение» с разными параметрами.

```
im = Image('background.jpg', 44.1, 'JPG', 800, 600)
im2 = Image('new_background.jpg', 42.8, 'JPG', 800, 600)
```

Обратимся к методам первого экземпляра.

```
im.get_amount()
im.kbs2bytes()
```

Теперь сравним два экземпляра класса «Изображение».

```
if (im == im2) is True:
    print(f'{im.get_name()} и {im2.get_name()} равны.')
else:
    print(f'{im.get_name()} и {im2.get_name()} не равны.')
```

В конце распечатаем количество экземпляров родительского класса «Файл», туда же войдут и все экземпляры дочернего класса.

```
print(f'Objects count: {File.objects_count}')
```

Задание и варианты

- Создать класс с полями, в котором реализовать инициализатор и метод обработки данных.
- Спроектировать иерархию классов от изначально написанного класса, используя наследование. Дописать как минимум одно уникальное поле для каждого класса.
- В классах-наследниках реализовать метод обработки данных.

Вариант 1

- *Класс и его поля*: Здание – площадь, стоимость 1 кв. метра, количество проживающих
- *Метод 1*: Рассчитать общую стоимость
- *Иерархия*: деревенский дом, многоквартирный городской дом
- *Метод 2*: Соотношение стоимости к числу проживающих

Вариант 2

- *Класс и его поля*: Станок - производительность (изделий в час), стоимость станка, средняя цена детали
- *Метод 1*: количество деталей для окупаемости
- *Иерархия*: фрезерный станок, станок с ЧП
- *Метод 2*: время окупаемости станка при фиксированной цене детали

Вариант 3

- *Класс и его поля*: Автомобиль: ёмкость бака, расход топлива, средняя скорость
- *Метод 1*: Рассчитать пройденное расстояние до полного опустошения бака
- *Иерархия*: грузовик, автобус
- *Метод 2*: Соотношение веса груза/числа пассажиров к количеству топлива на 250 км

Вариант 4

- *Класс и его поля*: Огнестрельное оружие: количество патронов в магазине, скорострельность, дальность стрельбы
- *Метод 1*: За сколько секунд магазин опустеет
- *Иерархия*: штурмовая винтовка, снайперская винтовка
- *Метод 2*: Соотношение скорострельности к дальности стрельбы

Вариант 5

- *Класс и его поля*: Оптический диск – ёмкость, обороты, размер лазерного пучка в нм
- *Метод 1*: Сколько байт считывает лазерная головка за 20 секунд (считать все дорожки равными длине окружности диска)
- *Иерархия*: CD, DVD
- *Метод 2*: среднее время одного оборота каждого вида диска

Вариант 6

- *Класс и его поля*: Фигура - три числа: a, b, c
- *Метод 1*: Объём V = a * b * c
- *Иерархия*: Тело с внутренней полостью (все параметры уменьшаются на d), массив из фигур.

– Метод *2*: Объём тела за вычетом пустоты (V – (a–d) × (b–d) × (c – d)), суммарный объём нескольких одинаковых фигур

Вариант 7

– *Класс и его поля*: Лампа - мощность излучения в Вт, потребление энергии, срок службы
– *Метод 1*: через сколько дней лампа перегорит при работе 8 часов в сутки
– *Иерархия*: лампа дневного освещения, прожектор
– *Метод 2*: соотношение мощности излучения к энергопотреблению

Вариант 8

– *Класс и его поля*: Книга - количество страниц, время чтения одной страницы, количество картинок
– *Метод 1*: время чтения книги
– *Иерархия*: энциклопедия, телефонный справочник
– *Метод 2*: количество статей/номеров на страницу

Вариант 9

– *Класс и его поля*: Персонаж видеоигры: очки здоровья, очки выносливости, урон
– *Метод 1*: Рассчитать, сколько ударов наносит персонаж, пока выносливость не кончится
– *Иерархия*: слабый враг, босс
– *Метод 2*: Рассчитать, сколько ударов понадобится до смерти врага

Вариант 10

– *Класс и его поля*: Сотрудник: кол-во рабочих часов, ставка, коэффициент премии
– *Метод 1*: Рассчитать размер премии
– *Иерархия*: старший сотрудник, директор
– *Метод 2*: Соотношение зарплаты к рабочим часам

Дополнительное задание

Перегрузите оператор сложения `add`.

Контрольные вопросы

1) Какие принципы лежат в основе объектно-ориентированного программирования, и как они реализуются в Python?
2) Что такое класс и объект в Python, и чем они отличаются друг от друга?
3) Как работает наследование в Python, и чем отличается множественное наследование от одиночного?
4) Что такое инкапсуляция, и как можно скрывать данные в классах Python?

5) В чем заключается полиморфизм в ООП, и как он проявляется при использовании методов в Python?

Литература

1) Брайн Уорд. Внутреннее устройство LINUX. СПб.: Питер, 2016—2019. 384 с.
2) Лутц М. Изучаем Python, 5-е издание. – Пер. с англ. – СПб.: Диалектика (Вильямс), 2023. – 1552 с.
3) Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.
4) Лутц М. Программирование на Python, том I, 5-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2019. – 992 с.
5) Лутц М. Программирование на Python, том II, 5-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2019. – 992 с.
6) Гэддис Т. Начинаем программировать на Python. – 4-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2019. – 768 с.
7) Лучано Рамальо Python. К вершинам мастерства. – М.: ДМК Пресс, 2016. – 768 с.
8) Свейгарт, Эл. Автоматизация рутиных задач с помощью Python: практическое руководство для начинающих. Пер. с англ. — М.: Вильямс, 2016. – 592 с.
9) Рейтц К., Шлюссер Т. Автостопом по Python. – СПб.: Питер, 2017. – 336 с.: ил. – (Серия «Бестселлеры O’Reilly»).
10) Любанович Билл Простой Python. Современный стиль программирования. – СПб.: Питер, 2016. – 480 с.: – (Серия «Бестсепперы O’Reilly»).
11) Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2019. – 161 с. – (Бакалавр. Прикладной курс). – ISBN 978-5-534-10971-9. – Текст: электронный // ЭБС Юрайт [сайт]. – URL: https://urait.ru/bcode/437489 (дата обращения: 13.02.2024).
12) Шелудько, В. М. Основы программирования на языке высокого уровня Python: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 146 с. – ISBN 978-5-9275-2649-9. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. – URL: http://www.iprbookshop.ru/87461.html (дата обращения: 13.02.2024). – Режим доступа: для авторизир. пользователей
13) Шелудько, В. М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 107 с. – ISBN 978-5-9275-2648-2. – Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. –

URL: http://www.iprbookshop.ru/87530.html (дата обращения: 13.02.2024). – Режим доступа: для авторизир. пользователей
14) Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
15) Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.
16) Пилгрим Марк. Погружение в Python 3 (Dive into Python 3 на русском)
17) Прохоренок Н.А. Самое необходимое. – СПб.: БХВ-Петербург, 2011. – 416 с.
18) Сергеева, О. А. Программирование на Python : учебно-методическое пособие / О. А. Сергеева. – Кемерово : КемГУ, 2024. – 157 с. – ISBN 978-5-8353-3123-9. – Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/420758 (дата обращения: 11.11.2024). – Режим доступа: для авториз. пользователей.
19) Никитина, Т. П. Программирование. Основы Python / Т. П. Никитина, Л. В. Королев. — Санкт-Петербург : Лань, 2023. – 156 с. – ISBN 978-5-507-45283-5. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/302714 (дата обращения: 11.11.2024). – Режим доступа: для авториз. пользователей.
20) Янцев, В. В. Web-программирование на Python : учебное пособие для вузов / В. В. Янцев. – 3-е изд., перераб. – Санкт-Петербург : Лань, 2024. – 180 с. – ISBN 978-5-507-48364-8. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/392993 (дата обращения: 11.11.2024). – Режим доступа: для авториз. пользователей.
21) Груздев, А. В. Изучаем Pandas / А. В. Груздев, М. Хейдт ; перевод с английского А. В. Груздева. – 2-ое изд., испр. и доп. – Москва : ДМК Пресс, 2019. – 700 с. – ISBN 978-5-97060-670-4. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/131693 (дата обращения: 11.11.2024). – Режим доступа: для авториз. пользователей.
22) Бизли, Д. Python. Книга рецептов / Д. Бизли, Б. К. Джонс ; перевод с английского Б. В. Уварова. – Москва : ДМК Пресс, 2019. – 646 с. – ISBN 978-5-97060-751-0. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/131723 (дата обращения: 11.11.2024). – Режим доступа: для авториз. пользователей.
23) Нуньес-Иглесиас, Х. Элегантный SciPy / Х. Нуньес-Иглесиас, в. д. Уолт, Х. Дэшноу. — Москва : ДМК Пресс, 2018. – 266 с. – ISBN 978-5-97060-600-1. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/116124 (дата обращения: 11.11.2024). – Режим доступа: для авториз. пользователей.

Приложение А

Исходный код программы для лабораторной работы №1

```
import time

def draw_line(offset=0, length=1, color=222):
    line = ' ' * length
    print(f'{" " * offset}\x1b[48;5;{color}m{line}\x1b[0m')

def romb():
    height = 15
    center = height // 2
    offset = height // 2
    step = 1
    length = 1

    print(height, center, offset)
    colors = [222, 157]

    while True:
        for color in colors:
            for line in range(height):
                draw_line(offset, length, color)

                if line < center:
                    offset -= step
                    length += step * 2

                else:
                    offset += step
                    length -= step * 2

            print(f'\x1b[{height + 2}A')
            print(f'\x1b[{offset}D')

            length = 1
            offset = height // 2

            time.sleep(2)


if __name__ == '__main__':
    romb()
```

Приложение Б

Исходные коды программ для лабораторной работы №2

Ссылка на файл `memes_dataset.csv`: https://disk.yandex.ru/d/iDKYL2Hk7ksrsg

```
import json
import csv


DATASET_PATH = 'memes_dataset.csv'
OUT_PATH = 'out.json'


def get_title(dataset):
    dataset.seek(0)

    title = next(dataset)
    title = title.split(',')
    title = [col.strip() for col in title]

    print(title)
    return title


def get_object(line, title):
    fields = []
    value = ''
    in_complex = False

    for char in line:
        if in_complex:
            value += char

            if char == '"':
                value = value[:-1]
                fields.append(value)
                value = ''
                in_complex = False
        else:
            if char not in [',', '"']:
                value += char
                continue
```

```
            if char == ',':
                fields.append(value)
                value = ''
                continue

            if char == '"':
                in_complex = True
                continue

    result = {col: f for col, f in zip(title, fields)}
    return result

def get_object_alt(line, title):
    reader = csv.DictReader(
        [line],
        title,
        delimiter=',',
        quotechar='"'
    )
    res = next(reader)
    return res

def filter_year(dataset, title, year):
    filtered = []

    for line in dataset:
        obj = get_object(line, title)
        year_value = obj['origin_year']
        if year_value == str(year):
            filtered.append(obj)

    dataset.seek(0)
    return filtered

if __name__ == '__main__':
    with open(DATASET_PATH) as dataset:
       title = get_title(dataset)
       res = filter_year(dataset, title, 2008)
       res = json.dumps(res, indent=4)
       line = next(dataset)
       res = get_object_alt(line, title)
       with open(OUT_PATH, 'w') as out:
           out.write(res)
```

Приложение В

Исходный код программы для лабораторной работы №3

Файл `main.py`

```
import tkinter as tk
from chat import connector, reader, send_msg, read_msg
import threading


MSG_HEIGHT = 2
MAX_MSGS = 10

user_name = 'Anonymous'
msgs = []
msg_lbls = []


def  init_gui():
    root = tk.Tk()
    return root


def init_input(fr_list, fr_input):
    txt_entry = tk.Entry(fr_input)
    label = tk.Label(text='Никнейм')
    name_entry = tk.Entry()
    name_entry.insert(0, user_name)

    def click(txt_entry):
        text = txt_entry.get()
        msg = {
            'user':name_entry.get(),
            'text': text
        }
        send_msg(msg)
    btn_send = tk.Button(
        fr_input,
        text='Send',
        command= lambda: click(txt_entry)
    )

    txt_entry.pack(side=tk.LEFT, fill=tk.X, expand=True)
```

```
    btn_send.pack(side=tk.LEFT)
    label.pack(side=tk.LEFT)
    name_entry.pack(side=tk.LEFT)


def add_label (text):
    lbl_msg = tk.Label(
        fr_list_msg,
        text=text,
        height= MSG_HEIGHT,
        background='#92c2f2'
    )
    lbl_msg.pack(side=tk.TOP, anchor='ne', padx=2, pady=2)
    msg_lbls.append(lbl_msg)

    print(len(msg_lbls))

    if len(msg_lbls) > MAX_MSGS:
        label = msg_lbls.pop(0)
        label.destroy()


def init_frames(root):
    fr_list_msg = tk.Frame(root)
    fr_input_msg = tk.Frame(root)
    fr_list_msg.pack(fill=tk.BOTH, expand=True)
    fr_input_msg.pack(fill=tk.X, side=tk.BOTTOM)

    lbl_greet = tk.Label(
        fr_list_msg,
        text="Chat started..",
        height=MSG_HEIGHT
    )
    lbl_greet.pack(side=tk.TOP)

    return fr_list_msg, fr_input_msg


if __name__ == '__main__':
    root = init_gui()
    fr_list_msg, fr_input_msg = init_frames(root)
    th = threading.Thread(
        target=read_msg,
        args=(add_label, )
```

```
    )
    th.start()
    init_input(fr_list_msg, fr_input_msg)
    root.mainloop()
```

Файл `chat.py`

```
import redis
import json

HOST = 'de.futoke.ru'
PORT = 6379
PIPE_NAME = 'itmo'


def init_chat():
    connector = redis.StrictRedis(
        host=HOST,
        port=PORT,
        db=0,
        username='default',
        password='itmoredis',
        charset='utf-8',
        decode_responses=True
    )

    reader = connector.pubsub()
    reader.subscribe(PIPE_NAME)

    return connector, reader


def send_msg(msg):
    connector.publish(PIPE_NAME, json.dumps(msg))


def read_msg(callback):
    print('Start thread')

    for msg in reader.listen():
        print(msg)
        data = msg['data']
        if type(data) is int:
```

```
            continue

        else:
            data = json.loads(data)
            callback(f"{data['user']}:\n{data['text']}")

connector, reader = init_chat()
```

Приложение Г

Исходный код программы для лабораторной работы №4

```
MAX_VOLUME = 4


items = {
    'a': {'price': 700, 'volume': 3},
    'b': {'price': 300, 'volume': 1},
    'c': {'price': 500, 'volume': 2},
    'd': {'price': 550, 'volume': 2},
}


def gen_table(items, max_volume=MAX_VOLUME):
    table = [
        [0 for c in range(max_volume)] for _ in range(len(items))
    ]

    for i, (_, value) in enumerate(items.items()):
        price =  value['price']
        volume = value['volume']

        for limit_volume in range(1, max_volume+1):
            col = limit_volume - 1

            if i == 0:
                table[i][col] = 0 if volume > limit_volume else price
            else:
                prev_price = table[i-1][col]
                if volume > limit_volume:
                    table[i][col] = prev_price
                else:

                    used = 0 if col < volume else table[i-1][col-volume]

                    print(f'used {used} i {i} col{col} ')
                    print(f'{volume} {col-volume}')

                    new_price = used + price
                    res = max(new_price, prev_price)

                    table[i][col] = res
```

```
                print(f' col {col}, i {i}, table {table}, ')
                print(f'prev {prev_price}, new_p {new_price} ')

    print(table)

if __name__ == '__main__':
    gen_table(items)
```

Приложение Д

Шаблон оформления отчета по лабораторной работе

ИТМО

ЛАБОРАТОРНАЯ РАБОТА № __

По дисциплине:
«Программирование на Python»

Выполнил студент: Фамилия И.О.
Группа: X9999
Преподаватель: Фамилия И.О.

Санкт-Петербург
20XX

Цель работы

…

Задачи, решаемые в работе

…

Теоретическая часть

…

Экспериментальная часть

…

Выводы

Крылова Анастасия Андреевна
Шорохов Сергей Александрович
Афанасьев Максим Яковлевич
Федосов Юрий Валерьевич

Операционные системы семейства Linux

Лабораторный практикум

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49