

ИНФОРМАТИКА

7 - 9 класс

Фонд «Сорос-Кыргызстан»
Бишкек 2019

УДК 373.167.1
ББК 73 Я 721
И 74

Информатика: 7-9 класс. Учебное пособие для 7-9 классов общеобразовательных учреждений с русским языком обучения / И.Н. Цыбуля, Л.А.Самыкбаева, А.А. Беляев, Н.Н. Осипова, У.Э. Мамбетакунов; 1-ое изд. – Фонд Сорос-Кыргызстан, 2019 – 205 с.

ISBN 978-9967-31-911-0

Данное учебное пособие предназначено для учащихся 7-9 классов общеобразовательных школ, а также для детей любого возраста, готовых начать изучать основы информатики и программирования. Книга поможет им понять, как устроены компьютеры, как они взаимодействуют между собой и как ими управлять с помощью программ. Темы учебника раскрываются в четырёх основных разделах изучения информатики: Информатика и информация, Компьютеры и ПО, Компьютерные сети и интернет и Программирование. Все разделы повторяются в каждом классе, с постепенным расширением объема и глубины изучения материала. Учебник может быть использован как при работе в рамках школьной программы, так и при самостоятельном изучении языка программирования Python.

Для широкого круга читателей

Информатика. 7-9 класс

Э-версия книги размещена на сайте *bizdin.kg*

Технические эксперты: Али Палитаев, Исабек Ташиев, Санжар Маматов, Айдай Сабырова

Литературный редактор: Диана Светличная

Арт-директор: Мария Казакова

Настоящая книга разработана при поддержке **Фонда «Сорос-Кыргызстан»** под открытой лицензией Creative Commons Attribution 4.0 «С указанием авторства» (CC-BY) и является открытым образовательным ресурсом.



Данная лицензия позволяет третьим лицам свободно распространять, создавать производные (ремиксы, переводы), перерабатывать, адаптировать, в том числе и в коммерческих целях, всю книгу или любые ее части с обязательной ссылкой на ее авторов.

Более подробная информация об условиях данной лицензии представлена на сайте
<https://creativecommons.org/>

И 4306022200-19
ISBN 978-9967-31-911-0

УДК 373.167.1
ББК 73 Я 721



Фонд «Сорос Кыргызстан», 2019

ВВЕДЕНИЕ

Дорогие друзья!

Перед вами учебник, с помощью которого вы получите необходимые знания в области информационных технологий – важнейшей составляющей современной жизни.

Все мы являемся частью информационного общества, в котором информация – такой же ценный ресурс, как нефть и газ. Ваша задача – научиться правильно использовать этот ресурс, получать от него максимум пользы.

Этот учебник познакомит вас с технологией создания видеофильмов и веб-страниц, принципами кодирования информации и компьютерной графикой. Вы узнаете, как создавать электронные таблицы, презентации и базы данных, научитесь настраивать локальную сеть Wi-Fi и программировать на языке Python.

Вы иначе посмотрите на себя и на происходящее вокруг, вам откроется много нового и интересного! Современные информационные технологии, которые вы освоите, существенно изменят ваше представление об окружающем мире.

Этот учебник составлен так, чтобы вы сами могли оценивать уровень своего прогресса: в конце каждой темы вам предложены вопросы и задания для проверки теоретической базы, а также приведены компьютерные практикумы, с помощью которых вы сможете развить свои практические навыки.

В учебнике встречаются следующие условные обозначения:



ЗАПОМНИ

– важная информация, которую надо хорошо запомнить.



ЭТО ИНТЕРЕСНО!

– дополнительные сведения по теме.



ОПРЕДЕЛЕНИЯ

– теоретические сведения, которые необходимо знать наизусть.



ВОПРОСЫ И ЗАДАНИЯ

к объяснительному тексту учебника для самоконтроля.



КОМПЬЮТЕРНЫЙ ПРАКТИКУМ

– задания для самостоятельного выполнения на компьютере.

Мир открывается смелым и любознательным – задавайте себе вопросы, ищите на них ответы, помните о самодисциплине, не бойтесь экспериментировать.

Пусть у вас все получится!

СОДЕРЖАНИЕ

7 КЛАСС

Введение

1

Информатика и информация

- 10 Компьютер в жизни человека
- 12 Информационные процессы и хранение информации
- 16 Кодирование текстовой информации

2

Компьютер и ПО

- 20 Виды и состав программного обеспечения
- 22 Электронные таблицы
- 27 Презентации

3

Программирование

- 32 Язык программирования
- 38 Типы данных и операции над ними
- 42 Условные операторы
- 45 Циклы while и for

4

Компьютерные сети и интернет

- 50 Сложные поисковые запросы
- 51 Конструкторы сайтов
- 54 Электронная почта и облачные сервисы

8 КЛАСС

1

Информатика и информация

- 60 Логические выражения и операции
- 63 Законы логики
- 66 Решение логических выражений

2

Компьютер и ПО

- 72 ПО и виды лицензий
- 75 Базы данных

3

Программирование

- 82 Сложные условия: and, or, not

- 84 Списки, кортежи и словари
- 87 Циклические алгоритмы
- 92 Вложенные условные операции и циклы
- 96 Функции
- 102 Массивы
- 107 Строки и операции с ними
- 113 Форматирование строк
- 115 Работа с графикой в Python

4

Компьютерные сети и интернет

- 122 Компьютерные сети
- 127 Виды интернет-протоколов
- 131 Каскадные таблицы стилей

9 КЛАСС

1

Информатика и информация

- 140 Информационная грамотность
- 143 Шифрование и электронно-цифровая подпись

- 146 Кодирование графической информации

2

Компьютер и ПО

- 154 Компьютерная графика
- 158 Введение в робототехнику

3

Программирование

- 166 Рекурсия
- 170 Алгоритмы обработки массивов
- 177 Сортировка списков
- 182 Матрицы

4

Компьютерные сети и интернет

- 188 Технологии будущего
- 192 Безопасность в цифровом мире
- 198 Приложения

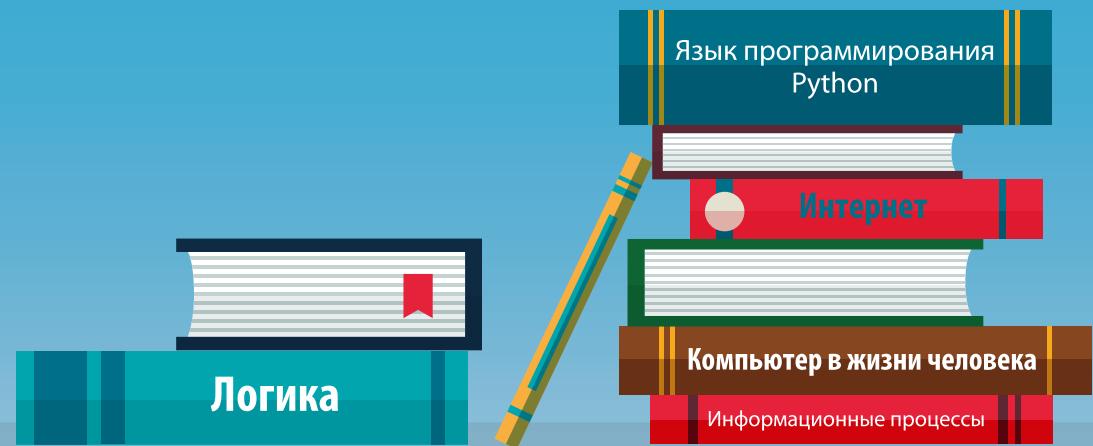
204

Глоссарий





класс



Логика

Глава 1



Информатика и информация

1.1. Тема:

Компьютер в жизни человека

Посмотрите вокруг – все, что нас окружает: от проектора в классе, холодильника дома, светофоров на улице, современных мультфильмов и онлайн игр до атомных станций и роботов на Луне, – создано и управляетя при помощи компьютеров.

Информационные технологии (ИТ) все глубже проникают в нашу повседневную жизнь, делая ее удобнее и безопаснее, а самое главное, помогают решать любые задачи гораздо быстрее. Сейчас в считанные секунды мы можем найти нужную информацию в онлайн энциклопедиях или отправить сообщение другу, находящемуся в другой стране.

Все более широко используются технологии на грани фантастики, такие как искусственный интеллект, виртуальная и дополненная реальность, биотехнологии для производства искусственных органов. Еще несколько лет назад трехмерная печать на принтере казалась фантастикой, а сейчас созданы принтеры, которые могут напечатать целый дом.

В то же время, в связи с глобальной компьютеризацией возникают вопросы о влиянии компьютера на физическое и психическое здоровье человека.

Длительная работа за компьютером приводит к изменениям в нервной, эндокринной, иммунной системах, негативно сказывается на зрении и костно-мышечном аппарате человека.

Вредные факторы, действующие на человека при работе с компьютером и другими электронными устройствами:



ОПРЕДЕЛЕНИЯ

Гиподинамия – низкая двигательная активность организма, снижение общей физической активности.

КОМПЬЮТЕР В ЖИЗНИ ЧЕЛОВЕКА

Положительные примеры развития технологий:

- 1** свободный доступ к открытым электронным энциклопедиям;
- 2** моментальная пересылка сообщений;
- 3** использование интернета для вызова такси, заказа билетов;
- 4** покупка вещей в онлайн магазинах;
- 5** использование искусственного интеллекта для прогнозирования;
- 6** 3D-печать реальных объектов.

Чтобы не нанести вред своему здоровью, работая за компьютером, нужно соблюдать правила:

- режим непрерывной работы за компьютером не должен превышать 20 мин;
- в перерывах можно сделать легкие упражнения;
- при появлении рези в глазах, резком ухудшении зрения, появлении боли в пальцах и кистях рук, усилении сердцебиения – немедленно покинуть рабочее место, сообщить о произошедшем учителю;
- расстояние от экрана до глаз – 50-70 см (расстояние вытянутой руки);
- шея слегка согнута;
- экран находится чуть ниже уровня глаз;
- спина прямая, плечи опущены и расслаблены;
- локти и колени находятся под прямым углом;
- край стула не давит на заднюю часть коленей;
- ноги располагаются прямо, не перекрецываясь.

ВОПРОСЫ И ЗАДАНИЯ:

1) Дополните таблицу положительного и отрицательного воздействия на жизнь человека:

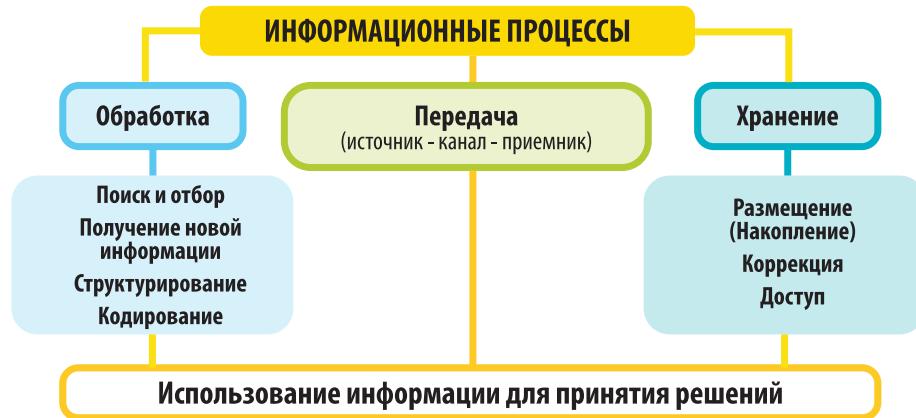
ПОЛОЖИТЕЛЬНОЕ ВОЗДЕЙСТВИЕ		ОТРИЦАТЕЛЬНОЕ ВОЗДЕЙСТВИЕ
1	Доступность общения	Компьютерная зависимость
2	Доступность любой информации	«Перегруз» ненужной информацией
3	Автоматизация работы	Сложность чтения с экрана
4	Возможность индивидуального обучения	Спам, фрагментарное чтение
5	Наглядность, возможность создавать презентации	Подмена реальности виртуальным миром

1.2. Тема:

Информационные процессы и хранение информации

Человек живет в мире информации. Каждый из нас хранит информацию в собственной памяти, а также в виде записей на различных внешних носителях: бумаге, магнитных, оптических. Процессы, связанные с получением, хранением, обработкой и передачей информации, называются **информационными процессами**.

Когда мы воспринимаем окружающий мир с помощью органов чувств – мы получаем *информацию*. В процессе общения с другими людьми – одновременно и *передаем*, и *принимаем информацию*. Чтобы ориентироваться в мире, мы запоминаем полученные сведения, то есть *храним информацию*. Для достижения каких-либо целей и принятия правильных решений мы *обрабатываем информацию*.



Измерение объема информации

По форме хранения информацию разделяют на аналоговую и цифровую. Разница между ними в том, что аналоговая информация непрерывна, а цифровая – дискретна. Для хранения любой информации необходимо измерить ее объем.

В информатике разделяют два основных подхода к измерению информации:

- 1 Содержательный подход
- 2 Алфавитный подход

Содержательный подход

Объем информации при таком подходе зависит от информативности полученного сообщения. Если сообщение не информативно (не несет определенного смысла для получателя сообщения), то объем информации будет равен нулю.

В этом случае информативность сообщения зависит от объема **полезной** в нем информации, такое сообщение снимает полностью или уменьшает неопределенность какой-либо ситуации.

Например, неопределенность информации о том, выходит ли человек на следующей остановке, заключается в выборе одного из двух вариантов ответа: «Да» или «Нет».

Для определения объема информации (i) для равновероятных событий используется формула $N = 2^i$, где N – это количество возможных событий.

Задача 1. У пассажира спросили: «Вы выходите на следующей остановке?» – «Да», – ответил он. Сколько информации содержит ответ?

Решение:

$$N = 2^i$$

N = 2, (количество возможных вариантов ответа: «Да» или «Нет»).

Тогда:

$$2 = 2^i \quad i = 1$$

Ответ: 1 бит

Пример 1. Книга лежит на одной из двух полок – верхней или нижней. Сообщение о том, что книга лежит на нижней полке, уменьшает неопределенность ровно вдвое и несет 1 бит информации.

Пример 2. В олимпиаде по информатике участвуют 4 ученика. Сообщение о том, что второй участник набрал большее количество баллов, уменьшает первоначальную неопределенность ровно в четыре раза (дважды по два) и несет два бита информации.

Алфавитный подход

Когда объем информации зависит от информационного веса символов, использованных в тексте на любом языке (естественном или формальном).

При алфавитном подходе используют понятие **мощность алфавита**, обозначающее полное число символов в алфавите, включая цифры и знаки препинания.

Например: мощность алфавита русских букв и используемых символов равна 54, т. е. 33 буквы + 10 цифр + 11 знаков препинания, скобки, пробел.

Наименьшую мощность имеет алфавит, используемый в компьютере (машинный язык), его называют двоичным алфавитом, т.к. он содержит только два знака «0» и «1». Информационный объем символа двоичного алфавита принят за единицу измерения информации и называется 1 бит.

Информационный объем одного символа алфавита в битах может быть вычислен по следующей таблице:

Количество вариантов (мощность алфавита)	2	4	8	16	32	64	128	256	512	1024
Количество бит информации	1	2	3	4	5	6	7	8	9	10

Таким образом, формула мощности алфавита будет выглядеть так:

$$M=2^K,$$

где **M** – мощность алфавита,

K – количество бит (информационный вес символа).

Для определения количества информации в сообщении (S) нужно умножить количество символов этого текста (N) на число бит для кодирования одного символа (K) данного алфавита:

$$S=N*K.$$

Задача 1. Алфавит содержит 32 буквы. Какое количество информации несет одна буква?

Решение:

Мощность алфавита $M = 32$.

1) $32 = 2^5$, значит вес одного символа $K = 5$ бит.

Ответ: 5 бит



ОПРЕДЕЛЕНИЯ

Естественный язык

используется для общения между людьми (русский, кыргызский, английский языки).

Формальный язык –

искусственный язык, характеризующийся точными правилами построения выражений: ноты, грамота, азбука Морзе, символы химических элементов, языки программирования.

Задача 2. Сообщение, записанное буквами из 64-символьного алфавита, содержит 20 символов. Какой объем информации в байтах оно несет?

Решение:

Мощность алфавита $M = 64$.

- 1) $64 = 2^6$, значит вес одного символа $K = 6$ бит.
- 2) 20 символов * 6 бит = 120 бит.
- 3) 120 бит : 8 = 15 байт.

Ответ: 15 байт

Задача 3. Одно племя имеет 32-символьный алфавит, а второе племя – 64-символьный алфавит. Вожди племен обменялись письмами. Письмо первого племени содержало 80 символов, а письмо второго племени – 70 символов. Сравните объем информации, содержащийся в письмах.

Решение:

Первое племя: $2^K = 32$, $K = 5$ бит – количество информации, которое несет каждый символ, $80 * 5 = 400$ бит.

Второе племя: $2^K = 64$, $K = 6$ бит – количество информации, которое несет каждый символ, $70 * 6 = 420$ бит.

Ответ: письмо второго племени содержит больше информации.

Задача 4. Сколько килобайт составляет сообщение, содержащее 12288 бит?

Решение:

1 килобайт = 1024 байт, 1 байт = 8 бит.

$12288 : 8 : 1024 = 1,5$ Кб.

Ответ: 1,5 Кб

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Сколько бит информации получено из сообщения «Бакыт живет на пятом этаже», если в доме 8 этажей?
- 2) Сколько килобайт составит сообщение из 384 символов 16-символьного алфавита?
- 3) Алфавит языка программирования содержит прописные и строчные буквы от A до Z и знаки арифметических операций. Какова мощность алфавита этого языка программирования?

1.3. Тема:

Кодирование текстовой информации

Любой текст состоит из символов, в том числе букв (заглавных или строчных), цифр, знаков препинания, спецсимволов, например «=», «(», «&» и т.п. и даже пробелов между словами. В памяти компьютера вместо символов хранятся их номера – числовые коды.

Стандартные кодировки текста

Кодовая таблица – таблица, в которой устанавливаются соответствия между числовыми кодами и символами. Изначально для кодирования одного символа использовался 1 байт (8 бит). Такая кодовая таблица содержит не более 256 символов (2^8). Для представления многоязычных текстов в 1991 году появился новый международный стандарт **Юникод** (англ. *Unicode*), в котором под 1 символ отводится 2 байта, что позволяет закодировать 65 536 символов. Полная спецификация стандарта Юникод включает в себя все существующие алфавиты мира. В 2017 году был принят стандарт Юникод, версия 10.0, который включает свыше 137 439 различных символов. В настоящее время используются многобайтовые кодировки (для представления одного символа необходимо несколько байт).

Таблица ASCII

ASCII (англ. *American Standard Code for Information Interchange*) – американский стандартный код для обмена информацией.

ASCII представляет собой кодировку для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляемых символов.

Коды с 0 по 32-ой – это специальные символы, которые содержат код перевода строки, пробела, ввода и т.д. Коды с 33 по 127 интернациональные и соответствуют буквам латинского алфавита, цифрам, знакам препинания, арифметическим операциям.



ОПРЕДЕЛЕНИЯ

Код – это система условных знаков и правил для представления информации.

Кодирование – это представление информации с помощью заданного кода.

Для отображения букв национальных алфавитов используют символы с кодами от 128 до 255. В национальных алфавитах одному и тому же коду соответствуют разные символы. Поэтому для правильного отображения текста требуется установить (выбрать в настройках программ) соответствующую ему кодовую страницу.

Задача 1. Текстовый файл содержит 32 страницы, на каждой странице 40 строк, в каждой строке 48 символов. Определите размер текста в кодировке КОИ-8, в которой каждый символ кодируется 8 битами.

Решение:

Найдем количество символов в статье: $32 \cdot 40 \cdot 48 = 61\,440$.

Один символ кодируется одним байтом. 10^{24} байт составляют 1 килобайт, поэтому информационный объем текста составляет 60 Кб.

Ответ: 60 Кб.

Задача 2. В одной из кодировок Unicode каждый символ кодируется 16 битами. Определите размер следующего предложения в данной кодировке: **Семь раз отмерь, один раз отрежь!**

Решение:

В предложении 33 символа. Следовательно, размер предложения в кодировке Unicode составляет: $33 \cdot 16 = \mathbf{528 \text{ бит}}$.

Ответ: 528 бит.

Для представления кириллицы существует несколько различных стандартов ASCII, в том числе: CP1251 (Windows), KOI-8 (UNIX), MacCyrillic (MacOS) и др. (См. Приложение 1)

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Что нужно сделать, если в браузере ваша страница отображается нечитаемым текстом?
- 2) Почему стандартной кодировки ASCII недостаточно для отображения всех алфавитов?
- 3) Можно ли отобразить с помощью стандартной кодировки ASCII текст на китайском языке? Поясните почему?
- 4) Определите, какой объем памяти займет следующая фраза, если используется кодировка KOI-8: *Молекулы состоят из атомов.*

Глава





Компьютер и ПО

2.1. Тема:

Виды и состав программного обеспечения

Традиционно программное обеспечение (ПО) разделяют на два вида: системное и прикладное ПО.

Системное ПО – это набор программ, которые управляют компонентами вычислительной системы, такими как процессор, коммуникационные и периферийные устройства.



Самой важной частью системного программного обеспечения является операционная система, которая выполняет следующие функции:

- распределяет и назначает использование ресурсов компьютера (процессор, ОЗУ, диски);
- планирует использование ресурсов компьютера и времени исполнения программ;
- осуществляет контроль работы компьютера.

В операционную систему может входить графический пользовательский интерфейс, обеспечивающий диалог человека и компьютера. Можно сказать, что операционная система является средой, в которой выполняются остальные программы.

Многозадачность – механизм, позволяющий выполнять на компьютере несколько задач одновременно: например, слушать музыку и работать в текстовом редакторе.

Система без поддержки многозадачности



Система с поддержкой многозадачности



В системах с поддержкой многозадачности компьютер используется более эффективно.

Механизм виртуальной памяти позволяет выделить часть внешней памяти (на жестком диске или другом носителе), чтобы в дальнейшем система рассматривала эту часть как продолжение ОЗУ. В результате компьютер может работать с большим количеством ОЗУ.

Прикладное программное обеспечение – это программы, обеспечивающие выполнение конкретных задач пользователя на компьютере.



Одним из наиболее распространенных прикладных ПО является набор свободных программ для работы с офисными документами OpenOffice.org, который позволяет работать с:

- текстовыми документами – OpenOffice.org Writer;
- электронными таблицами – OpenOffice.org Calc;
- слайдами презентаций – OpenOffice.org Impress;
- векторными рисунками – OpenOffice.org Draw;
- базами данных – OpenOffice.org Base;
- математическими формулами – OpenOffice.org Math.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Перечислите прикладные программы, которыми вы пользуетесь, с указанием их функций.
- 2) Какие системные программы обязательны для работы компьютера?
- 3) Каков алгоритм подключения нового принтера к компьютеру?

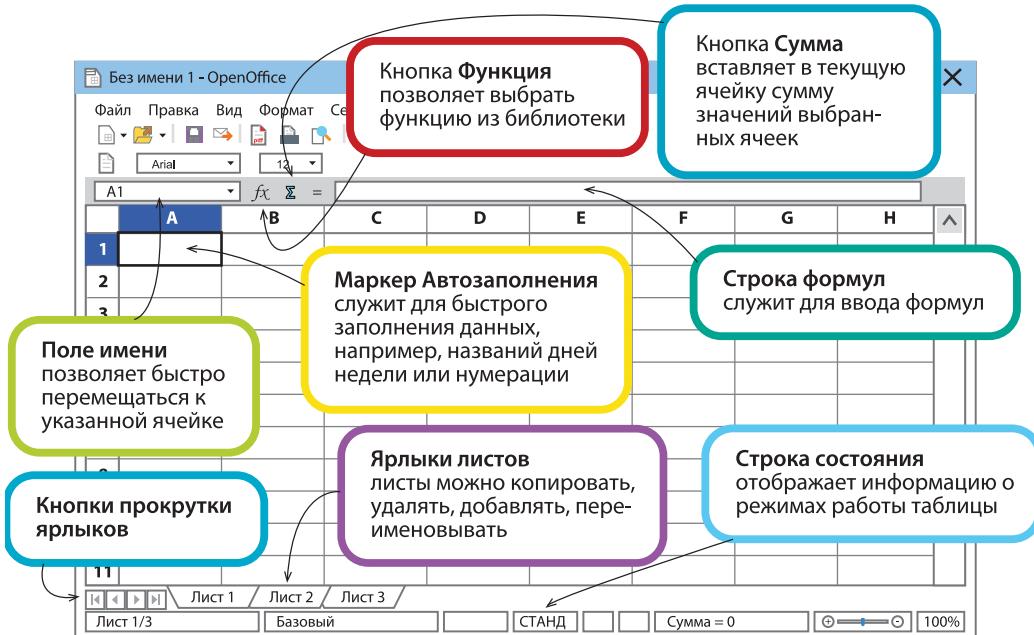
2.2. Тема:

Электронные таблицы

Для обработки, анализа и сортировки больших объемов числовых данных применяются электронные таблицы (ЭТ).

С их помощью вы можете:

- производить расчеты, объединяя данные, расположенные не только в разных ячейках, но и на разных листах;
- визуализировать информацию с помощью графиков и диаграмм.



Электронная таблица состоит из **Листов**, которые содержат:

Столбцы (обозначаются буквами латинского алфавита) и

Строки (обозначаются цифрами).

Пересечение строки и столбца образует **Ячейку**. Адрес активной ячейки указан в поле имени, например, A1.

В ячейках могут быть расположены числа, формулы и текст.

Группа ячеек образует **Диапазон**, он обозначается двоеточием, например, A1:B5. Способы выделения элементов ЭТ вы можете посмотреть в приложении 2.

Особенности ввода данных в электронных таблицах

Ввод данных производится в активную ячейку. Повторный ввод данных в эту ячейку удаляет ранее введенную информацию, поэтому для редактирования нужно нажать клавишу F2.

Если длина введенного текста превышает ширину ячейки, то будет отображена только часть строки.



ЭТО ИНТЕРЕСНО!

Если выделить несколько листов, вы можете добавлять и форматировать информацию на всех выделенных листах одновременно.

	A	B	C	D	E	F
1	Если соседние ячейки не заполнены, то строка	отображается полностью				
2	Если зап.	Строка отображается частично				
3	Можно использовать переносы	Можно растянуть ячейку				
4						

Для отображения информации целиком нужно изменить ширину ячейки или разрешить перенос строк.

Сортировка и фильтрация данных в электронных таблицах

 **A** Если в столбце расположены данные одного типа (тексты, числа или даты), можно произвести их сортировку по возрастанию или убыванию, используя кнопки на панели инструментов или команду  **Z**  **A** **Сортировка** в меню **Данные**.



Фильтр позволяет вывести на экран только те данные, которые соответствуют заданному условию. Установить фильтр для выбранного диапазона можно в меню **Данные** или с помощью кнопки на Панели инструментов.

Условное форматирование применяется для выделения данных, соответствующих заданному условию, с помощью изменения цвета, размера или гарнитуры шрифта. Для этого:

1. В меню **Формат** выберите команду **Условное форматирование**.
2. Примените условие к значению ячейки или формуле.
3. Установите стиль оформления.

Ввод формул

Ввод формулы всегда начинается со знака равенства «=», далее пишется сама формула. Например: =A4+16. К примеру, если в ячейке A4 мы записали число 20, то после записи в B4 формулы =A4+16 и нажатия *Enter*, в ячейке B4 появится число 36.

В строке формул отображается текущая формула, которую можно редактировать. Записывая сложные выражения, можно использовать знаки «+», «-», «*», «/», « \wedge ». Например: =A1+C5*B4

Относительные и абсолютные ссылки

В формулах используются ссылки на адреса ячеек. Ссылки разделяют на относительные, абсолютные и смешанные.

В относительных ссылках, при копировании формулы, ссылки на исходные данные изменяются (например, A1, B3).

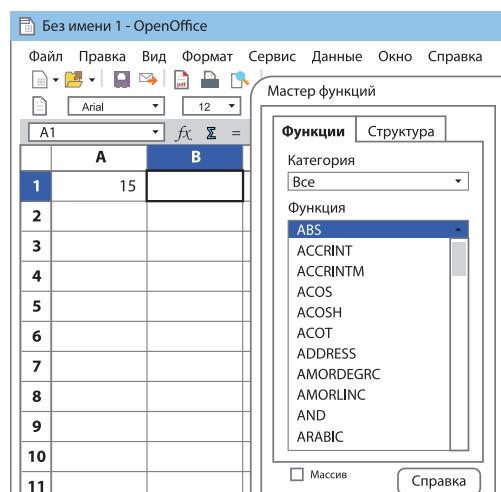
В абсолютных ссылках, при перемещении или копировании формулы, используется фиксированный адрес ячейки с исходными данными. Она обозначается с помощью знака доллара (например, \$A\$1).

В смешанных ссылках абсолютным остается адрес столбца (например, \$A1), или строки (например, A\$1).

Функции

В электронных таблицах доступен обширный набор функций следующих категорий:

- работа с базами данных;
- обработка времени и дат;
- финансовые;
- информационные;
- логические;
- математические;
- работа с массивами;
- статистические;
- текстовые;
- дополнительные.



Например, в категории «математические» вы можете выбрать функцию **СУММ** для подсчета суммы на выделенном диапазоне. В категории «статистические» вы можете выбрать функцию **СРЗНАЧ** для подсчета среднего значения на выделенном диапазоне.

Диаграммы

Для наглядного представления числовой информации, расположенной в электронных таблицах, используют диаграммы и графики.

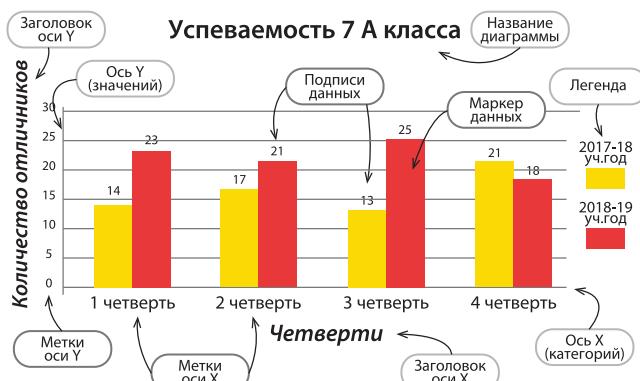
OpenOffice.Calc предоставляет разные типы диаграмм для решения определенных задач пользователя:

- Круговая
- Гистограмма
- Линейчатая
- Точечная и др.



Основные элементы диаграммы

- Область диаграммы
- Область построения
- Горизонтальная ось X - ось категорий
- Вертикальная ось Y - ось значений
- Ряд данных
- Легенда
- Название диаграммы
- Названия осей
- Подписи данных
- Точка данных



Пример создания диаграммы:

Расходы членов семьи за выходные

Имя	Суббота	Воскресенье
Папа	200	500
Мама	400	600
Брат	150	300
Сестра	300	200

Оформим таблицу расходов членов семьи за выходные по образцу.

Шаг 1. Для построения диаграммы необходимо выделить диапазон с исходными данными.

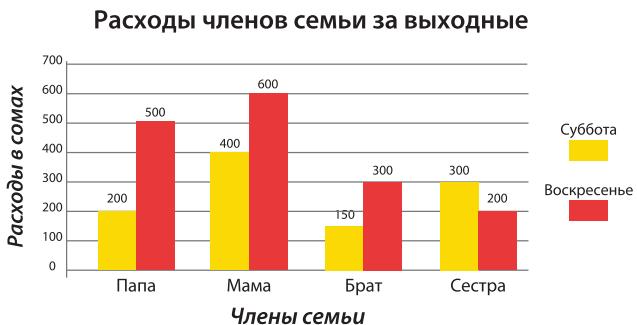
Шаг 2. В Меню *Вставить* выбрать команду *Диаграммы*.

Шаг 3. В открывшемся окне *Мастер диаграмм* указать место размещения диаграммы, ее тип (гистограмма) и название.

Шаг 4. Задать название рядов данных, выбрать отображение линий сетки, указать расположение легенды.

Получим диаграмму:

Для того чтобы изменить параметры диаграммы, достаточно щелкнуть ПКМ по изменяемому объекту и в контекстном меню выбрать необходимую команду.



Если изменить данные, на основе которых была построена диаграмма, то она будет перестроена автоматически.

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Создайте в таблице календарь на текущий учебный год.
- 2) Создайте таблицу Пифагора в электронной таблице.
- 3) Школьная команда баскетболистов состоит из 12 учеников. Постройте диаграмму роста учеников, если известны их имена и рост.
- 4) Рассчитайте еженедельную выручку музея, если известно:
 - количество проданных билетов каждый день;
 - цена взрослого билета – 40 сомов.
 - цена детского билета на 25% дешевле, чем взрослого.
 Постройте диаграмму (график) ежедневной выручки музея.
- 5) Используя функцию «ЕСЛИ», посчитайте, сколько каждый абонент должен заплатить за электроэнергию. При условии, что за первые 700 кВт/ч абонент платит 0,77 сома, а при превышении порога – стоимость 1 кВт/ч электроэнергии увеличивается до 2,16 сома.

	A	B	C	D	
1	№	Клиент	Количество электроэнергии	Оплата	
2	2	Асанова	135		
3	3	Давыдов	79		

2.3 Тема: Презентации

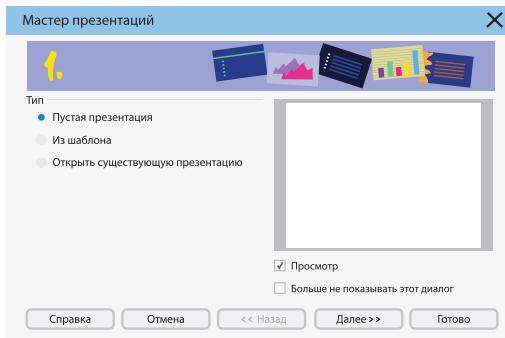


Программа OpenOffice Impress позволяет создавать презентацию с использованием текста, графических объектов, диаграмм, анимации, мультимедиа и других элементов.

Возможности OpenOffice.org Impress:

- создание слайдов на основе шаблонов с эффектами (анимация, эффекты переходов);
- создание слайдов на основе различных макетов;
- создание презентаций с использованием диаграмм;
- демонстрация презентаций в автоматическом или ручном режиме.

Создание презентации с помощью мастера презентаций



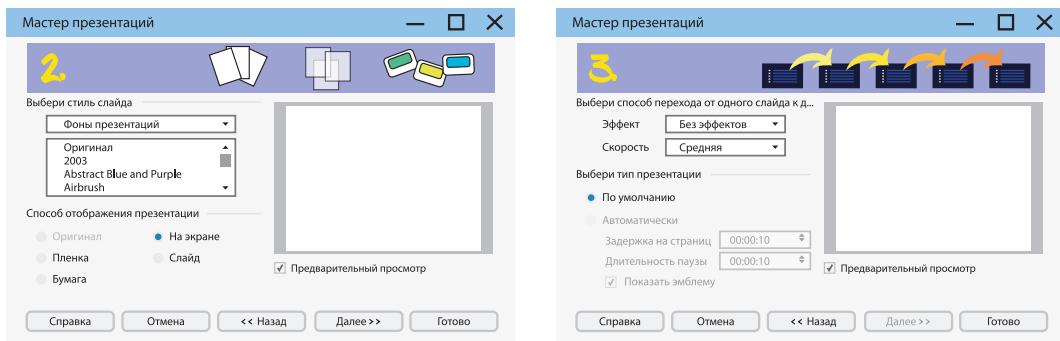
При запуске OpenOffice Impress на экране появляется окно «Мастер презентаций». Мастер презентаций помогает создавать презентации с помощью шаблонов, используя различные возможности для редактирования.

Шаг 1. В окне «Мастер презентаций» выберите тип презентации:

- пустая презентация – создает новую презентацию;
- из шаблона – создает новую презентацию с использованием шаблона;
- открыть существующую презентацию – открывает уже существующую презентацию.

Шаг 2. В следующем окне в группе «Выберите стиль слайда» в верхнем списке выберите один из двух стилей дизайна: презентацию или фон презентации.

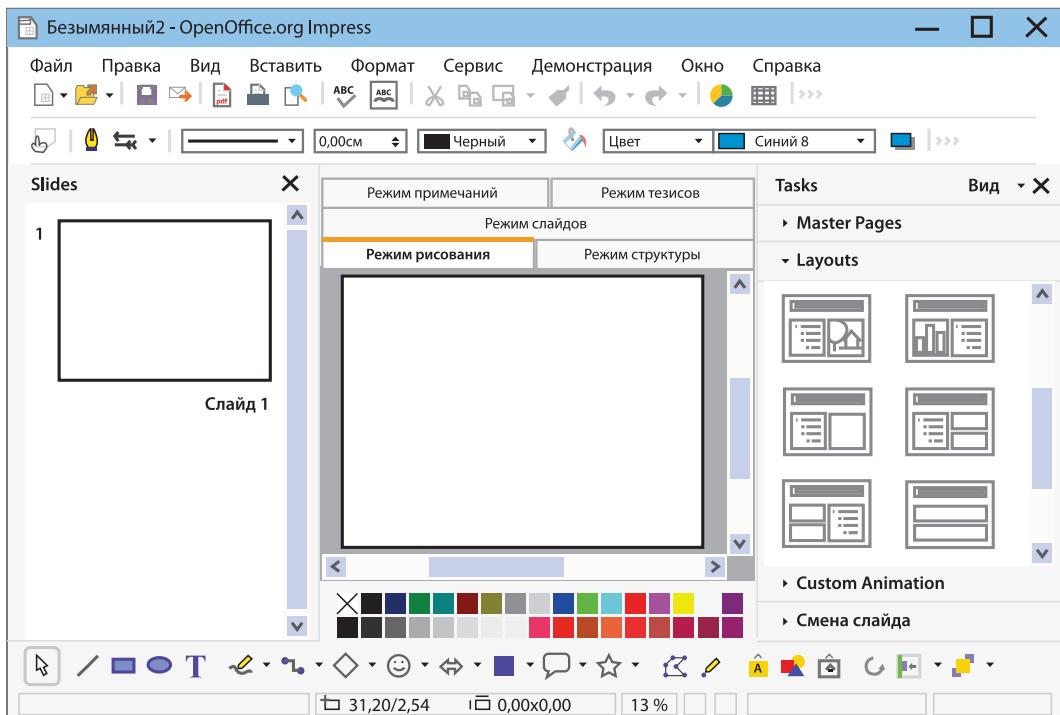
Шаг 3. В группе «Способ отображения презентации» выберите один из типов отображения: *Оригинал*, *Пленка*, *Бумага*, *На экране* или *Слайд* и нажмите на кнопку «Далее».



Шаг 4. В группе «Выберите способ перехода от одного слайда к другому», в списке «Эффект» выберите эффект для презентации. В списке «Скорость» установите скорость срабатывания эффекта, а в группе «Выберите тип презентации» определите время презентации.

На этом создание презентации с помощью мастера заканчивается. Для окончания необходимо щелкнуть по кнопке «Готово».

Перед вами откроется окно для форматирования презентации.



Форматирование слайдов

Форматирование текста в слайдах (изменение шрифта, размера, выравнивания и т.д.) и графических объектов (изменение размеров, яркости, обтекания и т.д.), производится по такому же принципу, как и в любом текстовом редакторе.

Для оформления титульного слайда установите макет «Заголовок, слайд» и введите заголовок презентации.

Для добавления следующего слайда в меню «Вставка» выберите команду «Слайд» или щелчком ПКМ на существующем слайде и выберите команду «Новый слайд» из контекстного меню.

Для добавления текстового поля в слайд необходимо щелкнуть мышью на пиктограмме  . После этого ЛКМ кликните на том месте слайда, куда необходимо добавить поле.

Для добавления графического объекта используют меню Вставка – Изображение – Из файла.

Презентация отличается от текстов возможностью добавления анимации. К графическим объектам и тексту можно добавить эффекты, которые можно выбрать на панели задач в группе «Эффекты».

Для демонстрации презентации нажмите кнопку Демонстрация  на панели инструментов Демонстрация или нажмите клавишу F5.

Для завершения демонстрации в любое время, в том числе и в конце, необходимо использовать клавишу Esc или щелкнуть мышью.

При сохранении презентации обратите внимание на тип файла. В том случае, если вы хотите чтобы ваша презентация открывалась в других программах, например, в Microsoft PowerPoint, необходимо в поле тип файла указать: Microsoft PowerPoint – версия – (.ppt) .

Для сохранения презентации в формате видео, нужно экспортовать вашу работу в другой формат, а для этого надо выбрать команду ЭКСПОРТ в меню ФАЙЛ.

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

Используя анимацию, создайте презентацию «История космонавтики». Сохраните ее в виде фильма.

Глава

3

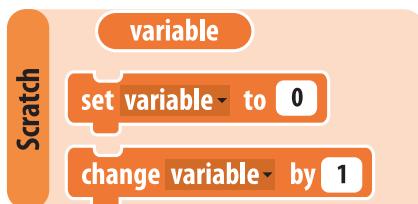


Программирование

3.1. Тема:

Язык программирования Python

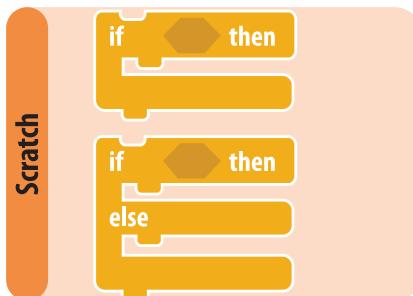
Язык программирования Python – это инструмент для создания программ самого разнообразного назначения, доступный даже для новичков. Если вы уже составляли программы из блоков в Scratch, то научиться писать программы на Python для вас будет намного легче. Давайте сравним, как выглядят команды в Scratch и в Python:



Python

```
variable = 0
variable = variable + 1
```

Так выглядят циклы:



Python

```
if True:
    #сделать что-то

if True:
    #сделать что-то
else:
    #сделать что-то
```

Python-код легко читается, а интерактивная оболочка позволяет вводить программы и сразу же получать результат.

На сегодняшний день на этом языке пишутся программы для банков, телекоммуникационных компаний, многие аналитики работают с данными с помощью именно этого языка. Благодаря понятному синтаксису на нем легко начать программировать.

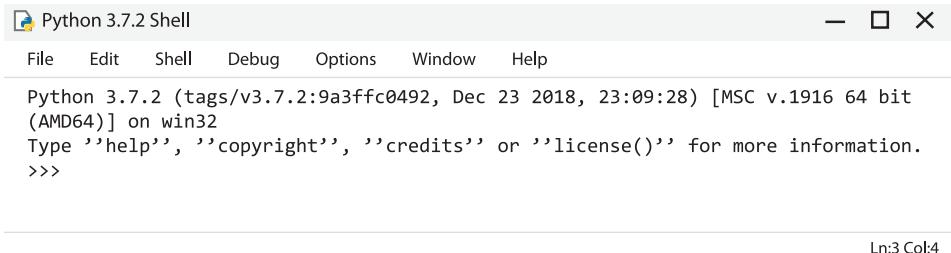
Для программирования на языке Python необходимо установить бесплатную среду программирования у себя на компьютере, которую можно загрузить по адресу: <https://www.python.org/downloads/>.

Вместе с Python на компьютер установится программа IDLE – среда разработки для написания Python-программ.

Для того чтобы записывать, сохранять и выполнять Python-команды, необходимо сделать следующие шаги:

1 Шаг. Запустить IDLE

Откроется окно консоли, в котором вы будете видеть результат вашей программы.



Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32

Type 'help', 'copyright', 'credits' or 'license()' for more information.

>>>

Ln:3 Col:4

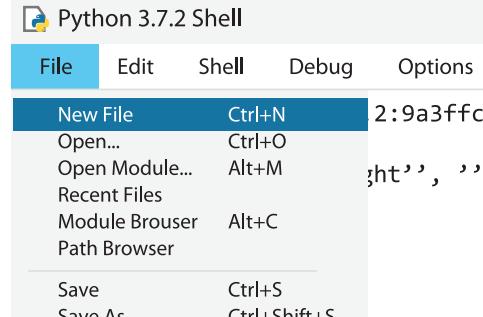
2 Шаг. Создать новый файл

Для записи новой программы необходимо создать отдельный файл. Для этого в меню **File** выберите **New File**:

3 Шаг. Ввести программу

В открывшемся окне введите вашу программу, например:

```
print('Salam')
```

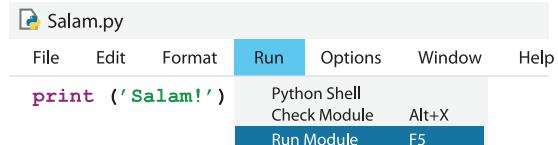


4 Шаг. Сохранить программу

В меню **File** выберите **Save As**, задайте новое имя для вашего файла и нажмите **Save**. Мы рекомендуем вам предварительно создать папку **My scripts** и сохранять туда все свои программы.

5 Шаг. Запустить программу

В меню **Run** выберите **Run Module** (либо можно просто нажать быструю клавишу F5)



Готово! В самой консоли вы должны увидеть сообщение:

>>>

Salam!

Способ, при котором программа сначала записывается в файл (обычно имеющий расширение **.py**) и при запуске исполняется целиком, называется программным режимом. Такой программный файл на Python называется скриптом (от англ. *script* – сценарий).

В Python также бывают пустые программы, которые не содержат ни одного оператора (команды). Часто это комментарии (пишутся после знака `#`) – пояснения, которые не обрабатываются интерпретатором.

#Это пустая программа

Функции в Python

В Python есть множество функций, при вызове которых программы выполняют содержащиеся в них команды.

print()

Функция **print** выводит информацию на экран. Она может выводить как значения переменных, так и значения выражений.

input()

Функция **input**, напротив, позволяет пользователю ввести данные для программы.

randint()

Функция **randint** возвращает случайное число из определенного диапазона.

*В скобках указывается значение, с которым будет работать эта функция.

Функция print выводит на экран символы, заключенные в апострофы или в кавычки. После выполнения этой команды происходит автоматический переход на новую строку.

Переменные

Напишем алгоритм программы, которая выполняет сложение двух чисел:

- 1) запрашивает у пользователя два целых числа;
- 2) складывает их;
- 3) выводит результат сложения.

ОПРЕДЕЛЕНИЯ

Переменная – это величина, которая имеет имя, тип и значение. Значение переменной может изменяться во время выполнения программы.

В условии появились данные, которые нужно хранить в памяти. Для этого используют переменные. Переменная (как и любая ячейка памяти) может хранить только одно значение. При записи в нее нового значения, предыдущее стирается и его уже никак не восстановить.

В языке Python переменные создаются в памяти при первом использовании, точнее, при присваивании им первоначального значения.

Например, при выполнении оператора присваивания:

```
a = 4
```

в памяти создается новая переменная под именем **a** (объект типа «целое число»). По этому имени теперь можно будет обращаться к переменной: считывать и изменять ее значение.

В именах переменных можно использовать латинские буквы (строчные и заглавные буквы различаются), цифры и знак подчеркивания «_».

Желательно давать переменным соответствующие имена, чтобы можно было сразу понять, какую роль выполняет та или иная переменная.



ЗАПОМНИ

Имя переменной не может начинаться с цифры, иначе транслятору будет сложно различить, где начинается имя, а где – число.

Тип переменной в Python определяется автоматически. Подробнее о типах переменных мы поговорим в следующей теме.

Для написания алгоритма нам нужно решить три подзадачи:

- 1 ввести два числа с клавиатуры и записать их в переменные (назовем их **a** и **b**);
- 2 сложить эти два числа и записать результат в третью переменную **c**;
- 3 вывести значение переменной **c** на экран.

Для ввода значения переменной используется оператор **input**:

```
a = input()
```

При выполнении этой строки система будет ожидать ввода с клавиатуры значения для переменной **a**. При нажатии клавиши *Enter* это значение запишется в переменную **a**. При вызове оператора **input** в скобках можно записать сообщение-подсказку:

```
a = input ('Введите целое число: ')
```

Сложить два значения и записать результат в переменную **c** очень просто:

```
c = a + b
```

ПРОГРАММИРОВАНИЕ

Символ «==» – это оператор присваивания, с его помощью изменяют значение переменной. Он выполняется следующим образом: вычисляется выражение справа от символа «==», а затем результат записывается в переменную, записанную слева. Поэтому, например, оператор

`i = i + 1`, увеличивает значение переменной `i` на 1.

Вывести значение переменной `c` на экран оператором `print`:

`print (c)`

Вся программа:

Программа на Python

```
a = input ('Введите целое число: ')
b = input ('Введите целое число: ')
c = a + b
print (c)
```

Результат вывода на экран

```
Введите целое число: 2
Введите целое число: 3
23
```

Мы видим, что два числа не сложились: программа просто объединила их, приписав вторую строку в конец первой. Потому что при такой записи введенные данные воспринимаются оператором `input` как символы, а не числа.

Чтобы исправить эту ошибку, нужно преобразовать символьную строку, которая получена при вводе, в целое число. Это делается с помощью функции `int` (от англ. *integer* – целый):

```
a = int (input())
b = int (input())
```

Возможен еще один вариант: оба числа вводятся не на разных строках, а в одной строке через пробел. В этом случае ввод выполняется сложнее:

```
a, b = map (int, input().split())
```

`map()` – применяет функцию ко всем элементам списка; в нашем случае это функция `int()`, которая превращает строку в целое число;

`split()` – строка разрезается на части по пробелам; в результате получается набор значений (список).

В результате после работы функции `map` мы получаем новый список, состоящий уже из чисел. Первое введенное число (первый элемент списка) записывается в переменную `a`, а второе – в переменную `b`.

Запишем программу с учетом рассмотренных функций:

Программа на Python

```
a, b = map(int, input().split())
c = a + b
print(c)
```

Результат вывода на экран

2 3
5

Теперь программа работает правильно – складывает два числа, введенных с клавиатуры. Однако у нее есть два недостатка:

- 1) перед вводом данных пользователь не знает, что от него требуется (сколько чисел нужно вводить и каких);
- 2) результат выдается в виде числа, которое неизвестно что означает.

Для того чтобы построить диалог программы с пользователем, программу можно записать так:

Программа на Python

```
print('Введите два целых числа: ')
a, b = map(int, input().split())
c = a + b
print(a, '+', b, '=', c, sep='')
```

Результат вывода на экран

Введите два целых числа:
2 3
2 + 3 = 5

Подсказку для ввода вы можете сделать самостоятельно.

При выводе результата нужно вывести значения трех переменных и два символа: «+» и «=», которые разделяются в операторе `print`:

```
print(a, '+', b, '=', c)
```

Для того чтобы убрать лишние пробелы, в операторе `print` используется так называемый разделитель (или сепаратор, англ. *separator*) – `sep`.

```
print(a, '+', b, '=', c, sep='')
```

Здесь мы установили пустой разделитель (пустую строку). В качестве разделителя можно указать любой знак. К примеру, если указать `sep = '*'` в команде:

```
print(1, 2, 3, 4, sep = '*')
```

то на экране мы увидим:

1*2*3*4*

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

Дан интервал времени в часах, минутах и секундах. Напишите программу, которая определит тот же интервал в секундах.

3.2. Тема:

Типы данных и операции над ними

Прежде чем использовать переменные, давайте разберемся с типами данных. Набор допустимых операций зависит от выбранного вами типа.

Типы данных

Перечислим основные типы данных в языке Python:

`int` – целые значения;

`float` – вещественные значения (числа с дробной частью);

`bool` – логические значения, `True` (истина, «да») или `False` (ложь, «нет»);

`str` – символ или символьная строка, то есть цепочка символов.

Целые переменные в Python могут быть сколько угодно большими (или, наоборот, маленькими, если речь идет об отрицательных числах): интерпретатор автоматически выделяет область памяти такого размера, который необходим для сохранения результата вычислений. Поэтому в Python легко точно выполнять вычисления с большим количеством значащих цифр.

При записи вещественных чисел в программе, целую и дробную часть разделяют не запятой, а точкой. Например:

`x = 123.456`

Логические переменные относятся к типу `bool` и принимают значения `True` (истина) или `False` (ложь).

Программа на Python

```
a = 10
b = 3
c = a/b
print ('c =', float (c))
```

```
a = 10
b = 3
c = a/b
print ('c =', int (c))
```

```
a = 10
b = 3
c = a/b
print ('c =', bool (c))
```

Результат вывода на экран

`c = 3.3333333333333335`

`c = 3`

`c = True`

Арифметические выражения и операции

В Python можно выполнять любые арифметические операции.

Арифметические выражения записываются в строчку. Они могут содержать числа, имена переменных, знаки арифметических операций, круглые скобки (для изменения порядка действий) и вызовы функций. Например:

```
a = (c + 5 - 1) / 2 * d
```

При определении порядка действий используется приоритет (старшинство) операций. Они выполняются в следующем порядке:

- 1 действия в скобках;
- 2 возведение в степень (**), слева направо;
- 3 умножение (*) и деление (/), слева направо;
- 4 сложение и вычитание, слева направо.

Равносильные записи выражений

```
a = b = 0
```

```
b = 0
```

```
a = b
```

```
a += b
```

```
a = a + b
```

```
a -= b
```

```
a = a - b
```

```
a *= b
```

```
a = a * b
```

```
a /= b
```

```
a = a / b
```

Нужно помнить, что результат деления (операции «/») – это вещественное число, даже если делимое и делитель – целые и делятся друг на друга нацело. Чтобы получить целый результат деления целых чисел, используют оператор «//». Чтобы выделить остаток от деления, используют оператор «%» (они имеют такой же приоритет, как умножение и деление):

Программа на Python

```
d = 85
a = d // 10 #=8
print (a)
b = d % 10  #=5
print (b)
```

Результат вывода на экран

```
8
```

```
5
```

Для отрицательных чисел:

```
print (-7 // 2)
print (-7 % 2)
```

```
-4
```

```
1
```

С точки зрения теории чисел, остаток – это неотрицательное число, поэтому $-7 = (-4)*2+1$, то есть частное от деления (-7) на 2 равно -4 , а остаток равен 1 .

В Python есть операция возведения в степень, которая обозначается двумя звездочками: « $**$ ». Например, выражение $y = 2x^2+z^3$ запишется так:

$$y = 2*x**2 + z**3$$

Задача 1. Напишите программу для вычисления площади треугольника, если известны его длина основания и высота.

Из геометрии мы знаем, что площадь треугольника равна произведению половины основания треугольника (a) на его высоту (h):

$$S = \frac{1}{2} ah$$

Эту формулу можно также записать так: $s=(a*h)/2$.

Теперь запишем программу и введем в качестве примера **6 см** как длину основания, а **4 см** как высоту треугольника.

Программа на Python

```
a = float(input('Введите значение основания: '))
h = float(input('Введите значение высоты: '))
s = (a*h)/2
print ('Ответ: s=' ,s)
```

Результат вывода на экран

```
Введите значение основания: 6
Введите значение высоты: 4
Ответ: s= 12.0
```

Случайные числа

Чтобы получить случайное целое число, сначала загрузим в Python функцию `randint`. Для этого используем команду `import` в окне консоли:

```
>>> from random import randint
>>> randint (1, 10)
7
```

Функция `randint()` выбрало случайное число в диапазоне от первого до второго числа в скобках. В нашем примере она выбрала цифру 7.

Стандартные функции

Большинство стандартных функций языка Python разбиты на группы по назначению, каждая группа записана в отдельный файл, который называется **модулем**. Математические функции собраны в модуле `math`.

Для подключения этого модуля используется команда импорта (загрузки модуля) – `import math`.

Ниже представлен некоторый функционал для работы с числами:

Команда	Выполняемое действие
<code>int(x)</code>	приведение вещественного числа x к целому, отбрасывание дробной части
<code>round(x)</code>	округление вещественного числа x к ближайшему целому
<code>ceil(x)</code>	округление до ближайшего большего числа
<code>floor(x)</code>	округление вниз
<code>abs(x)</code>	вычисление модуля числа (абсолютной величины)
<code>sqrt(x)</code>	квадратный корень числа x
<code>sin(x)</code>	синус x (x указывается в радианах)
<code>cos(x)</code>	косинус x (x указывается в радианах)
<code>tan(x)</code>	тангенс x (x указывается в радианах)

Для обращения к функциям используется точечная запись: указывают имя модуля и затем через точку название функции:

```
print (math.sqrt(x))
```

Программа на Python	Результат вывода на экран
<pre>x = 25 print (math.sqrt(x))</pre>	5.0

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

1) Даны температура в градусах Цельсия. Напишите программу для вычисления соответствующей температуры в градусах Фаренгейта. Формула перевода – $tF = 9/5 * tC + 32$

2) Напишите программу для вычисления значения выражения:

$$\frac{(a + b) * c + d * \frac{e}{f}}{k + p * \frac{b}{a} + g} * \frac{4}{5}$$

3.3. Тема:

Условные операторы

Возможности, которые мы рассмотрели до этого, позволяют писать **линейные** программы, в которых операторы выполняются последовательно друг за другом, и порядок их выполнения не зависит от входных данных.

В большинстве реальных задач порядок действий может несколько изменяться, в зависимости от того, какие данные поступили.

Из курса 6 класса вы помните, что если возникает выбор между двумя вариантами действий, то для записи такого алгоритма используется конструкция ветвления. Ветвление в Python реализуется с помощью условных операторов. В зависимости от значения, условные операторы направляют программу по одному из путей. Например, программа для системы пожарной сигнализации должна выдавать сигнал тревоги, если данные с датчиков показывают повышение температуры или задымленность.

Условный оператор `if` позволяет сначала проверить условие и только потом принять решение об исполнении или не исполнении дальнейших инструкций. Чтобы понять, как работает оператор `if`, рассмотрим типичные задачи на проверку условий и выбор.

Задача 1. Напишем программу, разрешающую доступ только для тех, чей возраст старше 21 года.

```
a = int(input('Введите свой возраст: '))
if a >= 21:
    print('Доступ разрешен')
else:
    print('Доступ запрещен')
```

Начало каждой «ветви» программы обозначается двоеточием «:». Условие в операторе `if` записывается без скобок.

Если условие, записанное после оператора `if` верно (истинно), то затем выполняются все команды (блок команд), которые расположены до другого условного оператора `else`. Если же условие после `if` неверно (ложно), выполняются команды, стоящие после `else`. В нашем примере, если указать возраст – 13, то исполнится инструкция, записанная после `else:`, то есть доступ будет запрещен.

В Python важную роль играют сдвиги операторов относительно левой границы (отступы). Обратите внимание, что слова **if** и **else** начинаются на одном уровне, а все команды внутренних блоков сдвинуты относительно этого уровня вправо на одно и то же расстояние. Для сдвига используют символы табуляции: одно нажатие на клавишу Tab или четыре пробела.

Если необходимо ввести несколько альтернативных условий, то можно использовать дополнительные блоки **elif** (сокращенное от **else – if**), после которого идет блок инструкций.

```
a=int(input('Введите свой возраст: '))
if a >= 21:
    print('Доступ разрешен')
elif a >= 18:
    print('Доступ частично разрешен')
else:
    print('Доступ запрещен')
```



ЗАПОМНИ

В Python в конце заголовков инструкций с условным оператором всегда ставится двоеточие.

Операторы сравнения

Операторы сравнения позволяют нам сравнить между собой два значения, где в результате выводится значение – либо True, либо False.

Математический символ	Оператор Python	Значение	Пример	Результат
<	<	Меньше, чем	1 < 2	True
>	>	Больше, чем	1 > 2	False
≤	≤	Меньше или равно	1 ≤ 2	True
≥	≥	Больше или равно	1 ≥ 2	False
=	==	Равно	1 == 2	False
≠	!=	Не равно	1 != 2	True

Задача 2. Компания проводит опрос общественного мнения и ее интересуют люди в возрасте от 20 до 70 лет включительно. Нужно написать программу, которая запрашивает возраст претендента и выдает ответ: «подходит» он или «не подходит» по этому признаку.

Пусть в переменной **v** записан возраст. Тогда нужный фрагмент программы будет выглядеть так:



```

if v >= 20 and v <= 70:
    print ('подходит')
else:
    print ('не подходит')

```

В языке Python разрешены двойные неравенства, например:

`if A < B < C:`
означает то же самое, что и

`if A < B and B < C:`

Задача 3. Запишем программу по сохранению файла с использованием условных операторов и операторов сравнения:

```

ans = input('Вы хотите сохранить файл? (да/нет)')
if ans == 'да':
    print('Выберите папку для сохранения')
if ans == 'нет':
    print('Данные будут утеряны для дальнейшего использования')
else:
    print('Ошибка. Такого варианта ответа нет')

```

Результат вывода будет зависеть от того, какой ответ был выбран: «да» или «нет»:

Вариант 1:

Вы хотите сохранить файл? (да/нет) да
Выберите папку для сохранения

Вариант 2:

Вы хотите сохранить файл? (да/нет) нет
Данные будут утеряны для дальнейшего использования

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Даны три числа. Напишите программу, которая выводит количество одинаковых чисел в этой цепочке.
- 2) Даны длины трех отрезков. Напишите программу, которая проверяет, могут ли они являться сторонами треугольника.
- 3) Дано натуральное число. Определите, будет ли это число: четным, кратным 4.
- 4) Напишите программу по обмену валюты в сомах, долларах и евро.



ЗАПОМНИ

= присваивает значение для переменной (если `a = b`, то `a` становится тем же, что и `b`);

== сравнивает два значения (если `a==b`, то это запрос на сравнение, и программа выдаст результат `True` или `False`).

3.4. Тема:

Циклы while и for

Циклы являются такой же важной частью программирования, как условные операторы. С их помощью можно организовать повторение некоторых частей кода. В Python для записи циклов используются 2 вида команд: **while** и **for**.

Цикл **while**

While переводится с английского как «пока», то есть цикл (блок команд) выполняется до тех пор, пока не выполнится заданное условие. Для этого в начале очередного шага цикла выполняется проверка условия. Поэтому оно называется циклом с предусловием.

Задача 1. Запишем программу для вывода на экран всех целых чисел от 1 до 5.

Программа на Python

```
d = 0
while d<5:
    d+=1
    print (d)
```

Результат вывода на экран

```
1
2
3
4
5
```

Проверяется предусловие так: если в начальный момент значение переменной **d** будет больше или равно 5, цикл не выполнится ни одного раза.

Задача 2. Рассмотрим пример: определите количество цифр в десятичной записи целого положительного числа. Будем предполагать, что исходное число записано в переменную **n** целого типа.

Для решения задачи нужно использовать переменную – **счетчик**, значение которой меняется с каждым новым проходом цикла. Для подсчета количества цифр нужно как-то отсекать эти цифры по одной, с начала или с конца, каждый раз увеличивая счетчик. Начальное значение счетчика должно быть равно нулю, так как до выполнения алгоритма еще не найдено ни одной цифры. Чтобы отсечь последнюю цифру, достаточно разделить число нацело на 10. Операции отсечения и увеличения счетчика нужно выполнять столько раз, сколько цифр в числе.

Как только результат очередного деления на 10 будет равен нулю, это и говорит о том, что отброшена последняя оставшаяся цифра. Программа на

Python выглядит так:

```
count = 0
while n > 0:
    n = n // 10
    count += 1
```

Количество проходов цикла будет равно количеству цифр введенного числа, то есть зависит от исходных данных. Если условие в заголовке цикла никогда не нарушится, цикл будет работать бесконечно долго. В этом случае говорят, что «программа зациклилась». Чтобы остановить зацикленную программу, нужно нажать на *Ctrl+C* в окне консоли.

Цикл `for`

Цикл `for` повторяет команды необходимое количество раз. Данная команда позволяет сделать программу компактнее. Рассмотрим предыдущий пример с использованием цикла `for`:

Программа на Python

```
for i in range (5):
    print (i)
```

Результат вывода на экран

```
0
1
2
3
4
```

Здесь переменная `i` (ее называют переменной цикла) изменяется в диапазоне (`in range`) от 0 до 5, не включая 5 (то есть от 0 до 4-х включительно). Таким образом, цикл выполняется ровно 5 раз.

Для того чтобы нам получить идентичный ответ (как в случае с `while`), изменим нашу программу:

Равносильные записи выражений

```
d = 0
while d < 5 :
    d+=1
    print ( d )
```

```
for i in range(1,6):
    print ( i )
```

Результат вывода на экран

```
1
2
3
4
5
```

Задача 2. Выведем степени числа два от 2^1 до 2^{10} (k = степени двойки).

Равносильные записи выражений

```
k = 1
while k <= 10 :
    print ( 2**k )
    k += 1
```

```
for k in range(1,11):
    print ( 2**k )
```

В первом варианте переменная `k` используется трижды: для присвоения начального значения, в условии цикла и в теле цикла (увеличение на 1).

Во втором варианте переменная `k` задается диапазоном (`range`) из двух чисел – начальным и конечным значением, причем конечное значение *не входит* в диапазон.

Шаг изменения переменной цикла по умолчанию равен 1. Если его нужно изменить, указывают третье (необязательное) число в скобках после слова `range` – это нужный **шаг**. Например, такой цикл выведет только нечетные степени числа 2:

Программа на Python

```
for k in range(1,11,2):  
    print ( 2**k )
```

Результат вывода на экран

```
2  
8  
32  
128  
512
```

С каждым шагом цикла переменная цикла может не только увеличиваться, но и уменьшаться. Для этого начальное значение должно быть больше конечного, а шаг – отрицательным. Следующая программа печатает квадраты натуральных чисел от 5 до 1 в порядке убывания:

Программа на Python

```
for k in range(5,0,-1):  
    print ( k**2 )
```

Результат вывода на экран

```
25  
16  
9  
4  
1
```

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Напишите программу, которая получает два целых числа A и B и выводит квадраты всех натуральных чисел в интервале от A до B .
- 2) Дано натуральное число. Напишите программу, которая находит сумму его цифр.
- 3) Дан бруск длиной 20 метров. Напишите программу, которая посчитает, какое минимальное целое количество отрезков длиной 1,5 м и 2 м получится из данного бруска.

Глава

Ч



Компьютерные сети и интернет

4.1. Тема:

Сложные поисковые запросы

Для улучшения результатов поиска информации в поисковых системах используются специальные символы или слова (операторы).

Например, найдем книги про животных и птиц автора Беляева.

1 животные & птицы & Беляев или +животные +птицы +Беляев

В искомом документе будут содержаться все заданные слова. При этом неважно, стоят они рядом или находятся в разных частях документа.

2 «животные и птицы Кыргызстана»

Отобразятся страницы с запросом в той форме, которую вы указали. Используется, если нужно найти точную цитату, название песни или фильма.

3 животные и птицы – Африка

Знак «минус» (–) перед запросом исключает слово из результатов поиска. «Минус» перед этим словом должен стоять без пробела.

4 животные|птицы или животные OR птицы

Оператор «OR» между двумя запросами позволит найти те страницы, где будет встречаться хотя бы одно из указанных слов.

5 животные и птицы site: www.kyrgyzstantravel.net

Этот оператор ограничивает поиск заданным доменом или сайтом.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Когда в критерии поиска надо задавать «+», а когда «-»?
- 2) Составьте сложный запрос на поиск информации по уходу за домашними кошками. Исключите из поиска крупных кошек (н-р, львов), а также предложения о покупке, продаже и т. п.
- 3) Для обозначения логической операции «ИЛИ» в запросе используется символ «|», а для логической операции «И» – «&». Расположите приведенные запросы в порядке возрастания количества страниц, которые нашел поисковый сервер и поясните свой ответ.

Код**Запрос****А**

Конан Дойль & Г. Бичер-Стоу & Джером К. Джером

Б

Конан Дойль | (Г. Бичер-Стоу & Джером К. Джером)

4.2. Тема:

Конструкторы сайтов

С тех пор как появился первый в мире сайт, технологии их разработки сильно изменились. Если изначально каждый сайт писался «с нуля», то с развитием интернета и бурным ростом количества веб-сайтов появились специальные конструкторы, позволяющие даже новичкам без знаний HTML верстки быстро создать свой сайт.

Эти конструкторы называются системами управления содержимым или «движком» сайта (content management system – CMS). С их помощью можно создавать и структурировать страницы сайта, редактировать содержимое на сайте: наполнять документами, метафайлами и пр.

В различных CMS, в большинстве случаев, есть готовое решение: это различные шаблоны сайтов-магазинов, сайтов-визиток, лэндингов (сайтов, призывающих к какому-либо действию). Но если вы задумали нетипичный сайт, тогда его рекомендуется писать с нуля.

По статистике, около 99% всех существующих сегодня веб-сайтов разработаны с использованием различных CMS, некоторые из них популярны по всему миру: WordPress, Drupal, Joomla и др.



ЭТО ИНТЕРЕСНО!

Первый в мире веб-сайт **info.cern.ch** был создан 6 августа 1991 года британским ученым-изобретателем Тимом Бернерсом-Ли. Это был простой текстовый веб-сайт без какой-либо графики, который объяснял, для чего нужна технология Word Wide Web.



ОПРЕДЕЛЕНИЯ

CMS – Система управления содержимым (контентом) (англ. Content management system, CMS) – это программное обеспечение, используемое для обеспечения и организации совместного процесса создания, редактирования и управления содержимым сайта.

Лэндинг (англ. landing page) – веб-страница, главной целью которой является совершение посетителем сайта конкретного (целевого) действия: например, проголосовать за что-то, ознакомиться с каким-либо товаром и купить его, заказать книгу.

Wix – платформа для создания сайтов

Одной из самых популярных платформ для быстрого создания небольшого сайта-визитки, лэндинга или блога является сервис Wix. В Wix есть множество готовых красивых шаблонов сайтов, где как в конструкторе вы можете добавлять или убирать лишние страницы, менять стиль, тексты, фоновые изображения, кнопки и др.

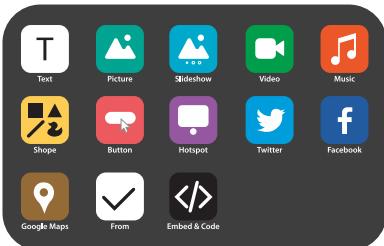
Перед началом создания веб-страницы вам необходимо определиться со структурой и контентом сайта и, в зависимости от этого, выбрать шаблон сайта.



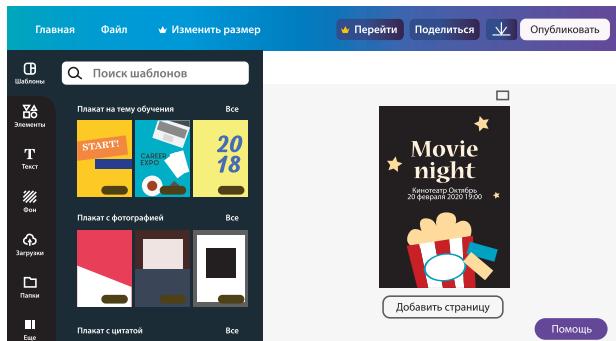
Наличие большого количества обучающих материалов в интернете позволит вам при желании изучить Wix более детально. Помните, что это – онлайн платформа, т.е. некоторые услуги более высокого уровня разработки являются платными.

Содержание сайта можно разнообразить, добавив много различного визуального контента, например, лонгриды и красивые презентации. В интернете есть много разных платформ для их создания.

Конструктор Readymag является одной из самых удобных онлайн платформ для верстки лонгридов. С его помощью к статье можно легко добавлять рисунки, слайд-шоу, видеофайлы, музыку.



readymag.com – платформа для создания лонгридов



ОПРЕДЕЛЕНИЯ

Лонгрид (англ. longread) – длинный текст (статья, рассказ), разбитый на блоки с помощью различных мультимедийных элементов: фотографий, картинок, видео, диаграмм и т.д.

Контент (англ. содержание) – это информация и опыт, предназначенные для конечного пользователя. Контент сайта – это все содержимое сайта, включая текст, картинки, видео и аудио.

Другой онлайн-сервис www.canva.com поможет вам быстро создать красивые открытки, постеры, баннеры для сайта, презентации, заставки для соцсетей. Все манипуляции с текстом, картинкой или фото выполняются прямо на сайте.

canva.com – сервис для дизайна графики

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

Создайте афишу Выпускного бала вашей школы с использованием онлайн платформ для дизайна графики.

4.3. Тема:

Электронная почта и облачные сервисы

Электронная почта – это один из наиболее часто используемых сервисов интернета.

Для работы с электронной почтой можно использовать либо почтовые клиенты, такие как Outlook Express, или же просто браузер.

Самым популярным почтовым сервисом в мире сегодня является Gmail. Регистрация личной учетной записи в Gmail (создание аккаунта Google) позволяет вам пользоваться всеми инструментами Google – электронной почтой, Диском, социальной сетью Google+, Play Market, YouTube сайтами и т.д. законно и бесплатно. Для того чтобы создать аккаунт Google, зайдите на сайт www.gmail.com и зарегистрируйте учетную запись.

При регистрации в анкете следует указывать ваше настоящее имя и реальную дату вашего рождения, так как искажение данных может привести к недоступности некоторых сервисов Google.

Необходимо использовать надежный пароль – сочетание букв в разных регистрациях, символов и цифр.

Адрес электронной почты записывается по определенной форме и состоит из двух частей, разделенных символом «@»:

Имя_адресата@доменное_имя_сервера

Имя_адресата – это имя пользователя (логин).

Приложения Google

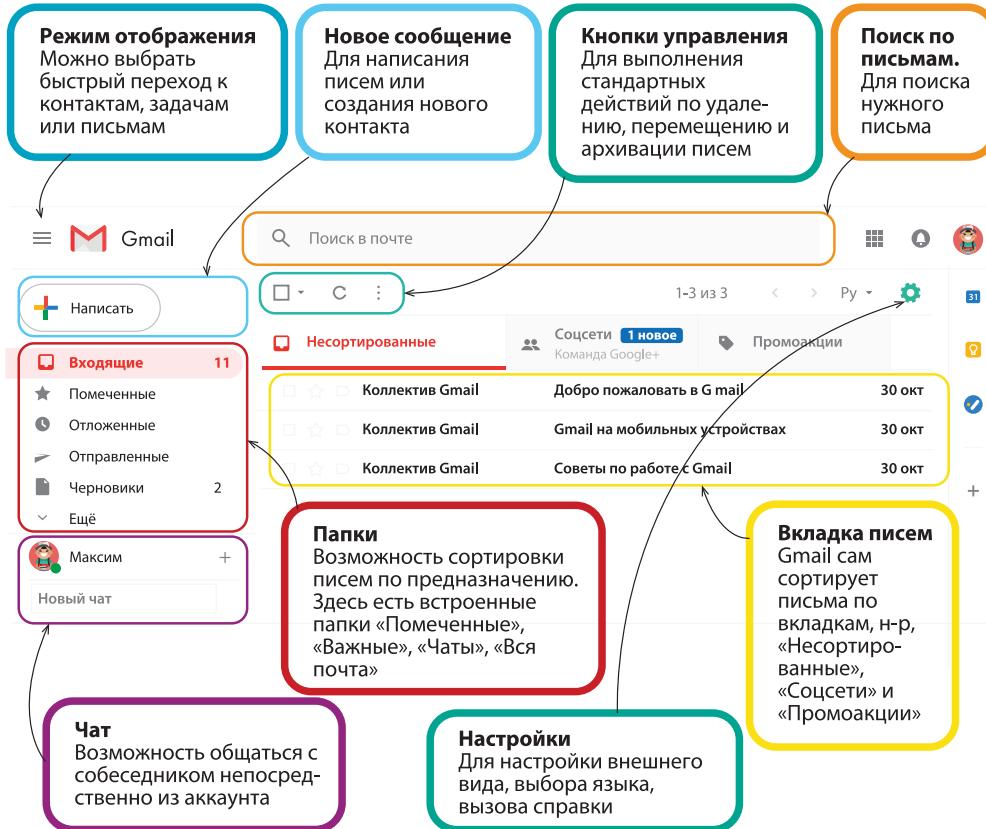
Google Disk – это сервис для хранения информации на виртуальном диске в сети интернет. В состав **Google Disk** входят приложения, которые позволяют создавать и редактировать онлайн текстовые документы, электронные таблицы, презентации и др. Владелец документа может предоставить доступ к совместной работе над документами и т.д.

Google Calendar – сервис для планирования встреч, где можно задавать время встречи и высыпать приглашение другим участникам по электронной почте.

Google Translate позволяет переводить слова, фразы и веб-страницы с одного языка на другой. Всего направлений языков около 1000, например, с русского на японский, сербский, хинди и наоборот.

ЭЛЕКТРОННАЯ ПОЧТА И ОБЛАЧНЫЕ СЕРВИСЫ

Окно почтового сервиса Gmail:

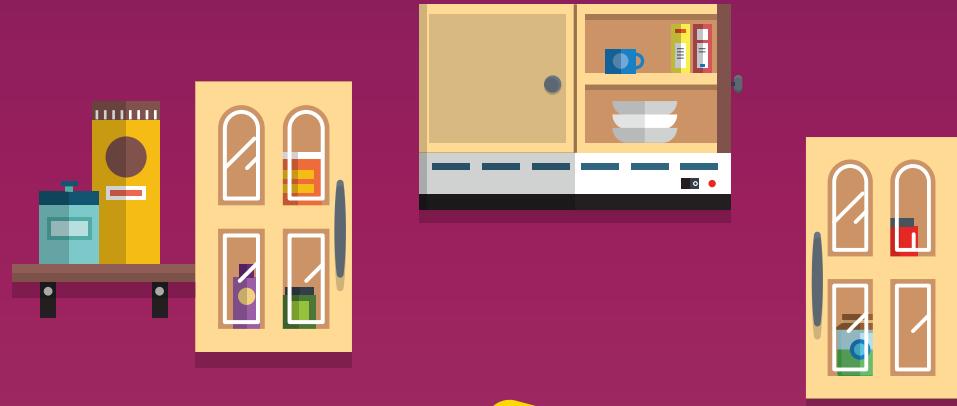


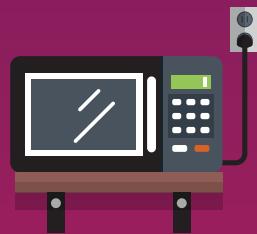
Google Photo – программа для работы с цифровыми фотографиями: обработка, создание коллажей и слайд-шоу фотографий.

Google Maps – карты и спутниковые снимки всего мира (включая Луну и Марс), а также карты автомобильных дорог с поиском маршрутов.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Создайте в Google-календаре напоминания о школьных каникулах и праздниках.
- 2) Найдите в Google maps три самых больших озера Кыргызстана и сделайте снимки их рельефных изображений.
- 3) Найдите в Google maps маршрут от Эйфелевой башни до Дома инвалидов в Париже и определите, сколько времени понадобится, чтобы добраться до места назначения пешком, на автомобиле и велосипеде.





класс



Глава 1



Информатика и информация

**1.1. Тема:**

Логические выражения и операции

В жизни мы часто используем слова «логично», «логика». В переводе с древнегреческого λογική – «наука о мышлении». Это наука о том, как правильно рассуждать, доказывать утверждения и делать выводы.

Логика изучает методы достижения истины не на основе чувственного опыта, а опираясь на ранее полученные знания. Логика применяется во многих науках, является подразделением математики, а булева алгебра является одной из основ информатики.



ЭТО ИНТЕРЕСНО!

В 1847 году английский математик Джордж Буль предложил применить для исследования логических высказываний математические методы, обозначая сложные словесные высказывания короткими символами. Позже этот раздел математики получил название **алгебра логики**, или «булева алгебра». С помощью нее записывают, вычисляют, упрощают и преобразовывают логические высказывания.

Булева алгебра очень проста, так как каждая переменная может принимать только два значения: истинно или ложно.

Алгебра логики, не вдаваясь в содержание высказываний, рассматривает только их истинность или ложность. Она позволяет определять истинность или ложность составных (сложных) высказываний путем алгебраических вычислений.

Логические высказывания обозначают заглавными буквами латинского алфавита:

A = {Земля круглая}.

B = {Солнце – это планета}.



ОПРЕДЕЛЕНИЯ

Простое логическое высказывание – это высказывание, которое нельзя уменьшить или разделить без потери смысла.

Сложные высказывания состоят из простых и соединены логическими связками, которые соответствуют логическим операциям.

Логические выражения – это логические высказывания, соединенные логическими операциями.

Основные логические операции

∨ Дизъюнкция (сложение) – это сложное логическое выражение, которое истинно, если хотя бы одно из простых логических выражений истинно, и ложно только тогда, когда оба простых логических выражения ложны. Логическое сложение образуется соединением двух (или более) высказываний в одно с помощью союза «или».

∧ Конъюнкция (умножение) – это сложное логическое выражение, которое истинно только в том случае, когда оба простых выражения являются истинными, во всех остальных случаях данное сложное выражение ложно. Логическое умножение образуется соединением двух (или более) высказываний в одно с помощью союза «и».

→ Импликация (следование) – это сложное логическое выражение, где первое выражение является условием, а второе является следствием. Такое логическое выражение истинно во всех случаях, кроме как из истины следует ложь.

Логическое следование образуется соединением двух высказываний в одно с помощью оборота речи «если..., то...».

¬ Инверсия (отрицание) – это сложное логическое выражение, если исходное логическое выражение истинно, то результат отрицания будет ложным, и наоборот, если исходное логическое выражение ложно, то результат отрицания будет истинным.

Логическое отрицание образуется путем добавления частицы «не» или слов «неверно, что» к исходному логическому выражению.

≡ Эквивалентность (равенство) – это сложное логическое выражение, которое является истинным только тогда, когда оба простых логических выражения имеют одинаковую истинность.

Логическое равенство образуется соединением двух высказываний в одно с помощью оборота речи «тогда и только тогда».

Основные логические операции и их аналоги в повседневной жизни

∨ = Чтобы поднять маме настроение, можно помыть посуду или предложить ей чаю.

∧ = Для достижения цели нужны знания и настойчивость.

→ = Если число делится на 6, то оно делится на три.

≡ = Пик Победы – самая высокая горная вершина в Кыргызстане.

**Таблица истинности для логических операций**

(Обозначим: 0 – это ложное высказывание, 1 – истинное)

Наименование операции	Сложение	Умножение	Отрицание	Следование	Равенство
Обозначение	\vee	\wedge	\neg	\rightarrow	\equiv
Таблица истинности	$0 \vee 0 = 0$ $0 \vee 1 = 1$ $1 \vee 0 = 1$ $1 \vee 1 = 1$	$0 \wedge 0 = 0$ $0 \wedge 1 = 0$ $1 \wedge 0 = 0$ $1 \wedge 1 = 1$	$\neg 0 = 1$ $\neg 1 = 0$	$0 \rightarrow 0 = 1$ $0 \rightarrow 1 = 1$ $1 \rightarrow 0 = 0$ $1 \rightarrow 1 = 1$	$0 \equiv 0 = 0$ $0 \equiv 1 = 0$ $1 \equiv 0 = 0$ $1 \equiv 1 = 1$
	или	и	не	если ... то	тогда и только тогда

ВОПРОСЫ И ЗАДАНИЯ:

1) Запишите на обычном языке сложные высказывания, выраженные формулами. Пусть A = «Тимуру нравится история», B = «Зое нравится информатика», C = «Рита может сделать видеоролик по истории».

а) $A \& B$; б) $B \rightarrow C$; в) $A \& C$; г) $(\neg A) \rightarrow (\neg C)$; д) $A \vee B \vee C$; е) $(A \& B) \rightarrow C$

2) Назовите целое число X , для которого данное высказывание является истинным: $HE(X > 6) \wedge (X > 4)$.

3) В таблице приведены запросы к поисковому серверу. Для обозначения логической операции «ИЛИ» в запросе используется символ \mid , а для логической операции «И» – $\&$: Для каждого запроса указан его код – соответствующая буква от А до Г. Расположите коды запросов слева направо в порядке возрастания количества страниц, которые нашел поисковый сервер по каждому запросу. По всем запросам было найдено разное количество страниц.

Код	Запрос
А	Эльфы Гномы Хоббиты Орки
Б	(Эльфы & Гномы) Орки
В	Эльфы & Гномы
Г	Эльфы Хоббиты

1.2. Тема:

ЗАКОНЫ ЛОГИКИ

Чтобы найти значение сложного логического выражения, нужно знать законы логики, некоторые из них похожи на законы арифметики:

ЗАКОН	ФОРМУЛИРОВКА
1. Закон тождества $A = A$	Всякое высказывание равно самому себе.
2. Переместительный закон (коммутативный) $A \wedge B = B \wedge A$ $A \vee B = B \vee A$	Результат операции над высказываниями не зависит от того, в каком порядке берутся эти высказывания. (От перемещения слагаемых сумма не меняется. От перемещения множителей произведение не меняется).
3. Сочетательный закон (ассоциативный) $(A \vee B) \vee Z = A \vee (B \vee Z)$ $(A \wedge B) \wedge Z = A \wedge (B \wedge Z)$	При одинаковых знаках скобки можно ставить произвольно или вообще опускать.
4. Распределительный закон (дистрибутивный) $(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$ $(A \vee B) \wedge C = (A \wedge C) \vee (B \wedge C)$	Определяет правило выноса общего высказывания за скобку.

Дополнительные законы:

Закон равносильности

$$A \vee A = A$$

$$A \wedge A = A$$

Закон поглощения

$$A \vee (A \wedge B) = A$$

$$A \wedge (A \vee B) = A$$

Закон общей инверсии**Закон де Моргана**

$$\neg(X \vee Y) = \neg X \wedge \neg Y$$

$$\neg(X \wedge Y) = \neg X \vee \neg Y$$

Закон непротиворечия

$$X \wedge \neg X = 0$$

Законы исключения констант

$$A \vee 1 = 1, A \vee 0 = A$$

$$A \wedge 1 = A, A \wedge 0 = 0$$

Закон склеивания

$$(A \wedge B) \vee (\neg A \wedge B) = B$$

$$(A \vee B) \wedge (\neg A \vee B) = B$$

Закон двойного отрицания

$$(\neg \neg X) = X$$

Закон исключенного третьего

$$X \vee \neg X = 1$$



Решение логических задач с помощью логических выражений осуществляется путем формализации.

Задача:

Ажара, Диана, Ира и **Тамара** изучают разные иностранные языки: английский, французский, китайский и японский.

Когда Андрей спросил у девочек, какой из языков изучает каждая из них,

Тамара ответила: «Я изучаю японский».

А остальные девочки сказали: «Это секрет, догадайся сам, если:

Ажара изучает китайский.

Диана не изучает китайский.

Ира не изучает французский.

При этом одно из трех утверждений истинно, а два других – ложны».

Поможем Андрею найти правильный ответ с помощью метода рассуждений.

Имеется 3 утверждения:

Запишем кратко:

Утверждение 1. **Ажара** изучает китайский.

A = *K*

Утверждение 2. **Диана** не изучает китайский.

D ≠ *K*

Утверждение 3. **Ира** не изучает французский.

I ≠ *Φ*

По условию – истинно из них только одно.

1. Предположим, что истинно первое утверждение:

«**Ажара** изучает китайский язык»

Тогда, по условию задачи, два других утверждения

A = *K* = 1

должны быть ложны. При этом, если ложно, что

D ≠ *K* = 0

Диана не изучает китайский, значит, она **изучает**

I ≠ *Φ* = 0

китайский.

Получается, что и **Ажара**, и **Диана** изучают китайский язык, а в условии сказано, что девочки изучают разные языки, следовательно, это предположение неверно и первое утверждение не может быть истиной.

A = *K* = 1

D ≠ *K* = 0

следовательно **D** = *K*

I ≠ *Φ* = 0

**2. Предположим, что истинно второе утверждение:
«Диана не изучает китайский».**

Тогда, по условию задачи, и первое, и третье утверждение ложны. Получается, что никто не изучает китайский язык. Значит, наше предположение ошибочно.

$$\text{А} = K=0$$

$$\text{Д} \neq K=1$$

$$\text{И} \neq \Phi=0$$

3. По условию задачи истинно одно из трех утверждений. Так как два первых были ложны, значит, истинно третье утверждение: «Ира не изучает французский», а первые два ложны.

$$\text{А} = K=0$$

$$\text{Д} \neq K=0$$

$$\text{И} \neq \Phi=1$$

Следовательно:

$\text{А} \neq K=1$ **Ажара** не изучает китайский (остается французский или английский).

$\text{Д} = K=1$ **Диана** изучает китайский.

Мы определили, что:

«**Диана** изучает китайский».

А какие языки изучают **Ира** и **Ажара**?

Так как «**Ира** не изучает французский», значит, она изучает английский.

Получается, что французский язык изучает **Ажара**.

Ответ:

Тамара изучает японский, **Диана** изучает китайский, **Ира** изучает английский, а **Ажара** изучает французский язык.



ВОПРОСЫ И ЗАДАНИЯ:

Индира, Эмиль и Лена нашли в земле старинный сосуд. Каждый высказал о нем по два предположения:

Индира: Это сосуд скифский и изготовлен в V веке.

Эмиль: Это сосуд согдийский и изготовлен в III веке.

Лена: Это сосуд не скифский и изготовлен в IV веке.

Учитель истории сказал ребятам, что каждый из них прав только в одном из двух предположений.

Где и в каком веке изготовлен сосуд?

**1.3. Тема:**

Решение логических выражений

Рассмотрим, как на основе задачи построить логическое выражение:

Представьте, что вам нужно запрограммировать систему сигнализации подводной лодки так, чтобы она давала аварийный сигнал, если вышли из строя два из трех двигателей.

- A** – «Первый двигатель вышел из строя».
- B** – «Второй двигатель вышел из строя».
- C** – «Третий двигатель вышел из строя».
- X** – «Аварийная ситуация».

Система сигнализации сработает в ситуации «X».

Состояние «X» можно записать в виде формулы:

$$X = (A \text{ и } B) \text{ или } (A \text{ и } C) \text{ или } (B \text{ и } C).$$

Мы выполнили формализацию.

С помощью символов логических операций это выражение будет записано так:

$$X = (A \vee B) \wedge (A \vee C) \wedge (B \vee C). \text{ Такой сигнал будет понятен компьютеру.}$$

Решение логических выражений с помощью алгебры высказываний:

Порядок выполнения логических операций:

- 1** Выражение, расположенное в скобках.
- 2** Отрицание.
- 3** Логическое умножение.
- 4** Логическое сложение.
- 5** Логическое следование.

Задача 1. Решение сложного логического выражения:

Даны простые выражения: $A=\{3=6\}$, $B=\{2<3\}$, $C=\{4<1\}$.

Определите истинность составного выражения: $(\neg A \vee A \wedge B) \wedge \neg C$

Алгоритм решения:

1) Определяем истинность простых выражений:

$A=\{3=6\}$ ложь (0)

$B=\{2<3\}$ истина (1)

$C=\{4<1\}$ ложь (0)

2) Подставляем значения в составное выражение:

$(\neg A \vee A \wedge B) \wedge \neg C = (\neg 0 \vee 0 \wedge 1) \wedge \neg 0$

3) Определяем значение выражения, расположенного в скобках, с учетом операции отрицания:

$(\neg 0 \vee 0 \wedge 1) = 1 \vee 0 \wedge 1$

4) Выполняем логическое умножение: $0 \wedge 1 = 0$

5) Выполняем логическое сложение: $1 \vee 0 = 1$

6) Получаем промежуточный результат:

$(\neg 0 \vee 0 \wedge 1) = 1$

7) Определяем значение исходного выражения:

$1 \wedge \neg 0 = 1 \wedge 1 = 1$ (истина)



Ответ: составное выражение: $(\neg A \vee A \wedge B) \wedge \neg C$ истинно.

Задача 2. Для какого из приведенных чисел истинно высказывание:

НЕ (число < 100) И НЕ (число четное)? 115, 108, 35, 8.

Пояснение: Запишем выражение в виде (число ≥ 100) И (число нечетное), проверим:

1) Истинно, так как истинны оба высказывания: 115 не меньше 100 и 115 – нечетное число. Это правильный ответ.

2) Ложно, так как ложно второе высказывание: 108 – четное.

3) Ложно, так как ложно первое высказывание: 35 не меньше 100.

4) Ложно, так как ложно второе высказывание: 8 – четное.



Построение таблиц истинности

Определить истинность логического выражения можно с помощью таблицы истинности.

Таблица истинности показывает, какие значения принимает сложное высказывание при всех возможных значениях, входящих в него простых высказываний.

Алгоритм решения логических выражений с помощью таблицы истинности:

- 1 подсчитать количество переменных n в логическом выражении;
- 2 определить число строк в таблице по формуле $m=2^n$, где n – количество переменных;
- 3 подсчитать количество логических операций в формуле;
- 4 установить последовательность выполнения логических операций с учетом скобок и приоритетов;
- 5 определить количество столбцов: число переменных + число операций;
- 6 выписать наборы входных переменных;
- 7 провести заполнение таблицы истинности по столбцам, выполняя логические операции в соответствии с установленной в пункте 4 последовательностью.

Пример: определить истинность выражения $\neg A \vee B$ с помощью построения таблицы истинности.

- 1 Определяем количество переменных в логическом выражении: 2.
- 2 Определяем число строк в таблице по формуле $m=2^n$: $2^2=4$.
- 3 Определяем количество логических операций в формуле: 2.
- 4 Устанавливаем последовательность выполнения логических операций: первая – логическое отрицание, вторая – логическое умножение.
- 5 Определяем количество столбцов в таблице: число переменных + число операций = $2 + 2 = 4$.

РЕШЕНИЕ ЛОГИЧЕСКИХ ВЫРАЖЕНИЙ

6

Выписываем наборы входных переменных.

7

Проводим заполнение таблицы истинности по столбцам, заполняя все возможные варианты значений переменных А и В.

8

Выполняем логические операции в соответствии с установленной в пункте 4 последовательностью.

$$\neg A \vee B$$

A	B	$\neg A$	$\neg A \vee B$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

ВОПРОСЫ И ЗАДАНИЯ:

1) Запишите логическое выражение на языке алгебры логики:

- На каникулах мы пойдем в театр или в цирк.
- 15 делится на 3 и на 5.
- Если сумма цифр числа делится на 3 без остатка, то число делится на 3.

2) Высказывание А «х – четное число», высказывание Б «х – делится на 5». Какой результат даст логическое сложение $A \vee B$?

3) Найдите значения логического выражения: $((1 \vee 0) \vee 1) \vee 0$

4) Даны простые высказывания:

$$A=\{3=6\}, B=\{2<3\}, C=\{4<1\}$$

Определите истинность составного высказывания:

$$(A \vee \neg B) \wedge \neg(\neg A \vee C).$$

5) Пусть a, b, c – логические величины, которые имеют следующие значения:

$$a=\text{истина}, b=\text{ложь}, c=\text{ложь}.$$

Составьте таблицу истинности для следующих логических выражений:

$$a) a \text{ или } b \text{ и не } c;$$

$$b) a \text{ и } b \text{ или } c \text{ и } b.$$

6) Докажите с помощью таблиц истинности изученные вами законы логики.

Глава

2



Компьютер и ПО

**2.1. Тема:****ПО и виды лицензий**

Вы уже убедились, что работа на компьютере невозможна без программного обеспечения (ПО). Оно может быть установлено по умолчанию (например, при установке операционной системы, в ее стандартный состав ПО входит простой текстовый редактор, калькулятор, графический редактор и т.д.) или дополняется по желанию пользователя.

Программное обеспечение**Проприетарное (несвободное) ПО**

Как известно, любая программа – это результат интеллектуальной деятельности человека, а значит – его собственность.

Например, вы приобрели в магазине сотовый телефон. Теперь это ваша собственность, а вот дизайн телефона, как и прежде, принадлежит разработчикам и дизайнерам, создавшим его. Если вы начнете производить копии этих телефонов, то нарушите авторские права разработчиков.

Такая же ситуация и с программами. Сама программа – это интеллектуальная собственность разработчика, а вы покупаете лишь результат работы этой программы. С точки зрения закона вы получаете только право на пользование программой. Все условия использования конкретной программы описаны в лицензионном соглашении, которое обычно появляется

Свободное ПО**ЗАПОМНИ**

Авторские права распространяются не только на программы, но и на фильмы, музыку и даже графические объекты (картинки, фото).

при установке программы. Соглашаясь с лицензионным соглашением, вы принимаете все условия разработчика программы. Если вы нарушаете хоть одно условие лицензионного соглашения (запрет на дальнейшее распространение, переделку и пр.), то ваша программа становится нелегальной.

История развития свободного ПО

В противовес развитию платного лицензионного (проприетарного) ПО в 80-х годах прошлого века зародилось движение среди программистов всего мира по развитию **свободного ПО (Open-source software)**.

Свободное ПО – это не только бесплатное ПО, но оно также имеет открытый исходный код, что позволяет желающим дорабатывать это ПО для своих нужд. Открытый исходный код распространяется на основе открытой лицензии.

Одной из таких открытых лицензий является лицензия **General Public License (GPL)**, создателем которой является основатель движения открытого ПО Ричард Столлман.

Особенностью использования такой лицензии является то, что любой пользователь имеет право распространять, изменять и дорабатывать такое ПО. Код всех новых изменений также должен быть открыт для других.

Под этой лицензией распространяется ядро Linux, MySQL, Asterisk и др. Второй популярной открытой лицензией для лицензирования ПО является **Apache License 2.0**. Эта лицензия позволяет закрывать часть кода, который был написан пользователем самостоятельно.



Большинство исходных кодов самой популярной операционной системы **Android** распространяются под свободной лицензией Apache 2.0. Сегодня на Android работают совершенно разные устройства, от «интернета вещей» и «умных домов» до телевизоров, ноутбуков и автомобилей, но чаще всего Android используют на смартфонах и планшетах.

Легальность использования информации из интернета

При пересылке файлов и использовании материалов из интернета, следует иметь в виду, что большинство информации в сети защищено законами об



Ричард Столлман



авторских правах (даже если на соответствующих веб-страницах об этом прямо не сообщается). Это означает, что вы можете только просматривать эту информацию, и не можете копировать ее, а тем более распространять и передавать другим. В большинстве случаев такая информация обозначена значком копирайта **© [имя автора]**. Он означает, что вам обязательно необходимо получить разрешение от владельца информации или веб-сайта на использование и переработку информации.

Поэтому прежде чем, например, скопировать иллюстрацию из сети в свой реферат или презентацию, следует убедиться, что данный материал разрешен автором для свободного использования другими пользователями сети. При этом надо понимать, что «бесплатное использование» это не значит «свободное использование». Бесплатное использование означает, что вы можете бесплатно просто смотреть и читать данный материал, но его нельзя передавать другим. Понятие «свободное использование» гораздо шире. Оно разрешает не только смотреть или читать данный материал, но также позволяет переделывать его (н-р, брать часть текста для доклада или использовать какие-то части из фильма), а также распространять дальше.

Образовательные материалы, свободно распространяемые в интернете, авторы которых разрешают другим пользоваться всеми правами на них, называются **открытыми образовательными ресурсами**. Самым крупным открытым образовательным ресурсом является Википедия, все статьи которой лицензированы открытой лицензией Creative Commons.

Эта лицензия позволяет авторам-создателям сообщать всем сразу, что его труд можно свободно использовать, распространять и дорабатывать. Даже этот учебник, который вы держите в руках, является открытым образовательным ресурсом и на первой ее странице вы можете найти значок открытой лицензии:



ВОПРОСЫ И ЗАДАНИЯ:

- 1) Почему компьютерное пиратство наносит ущерб обществу?
- 2) Приведите пример OpenSource программ и опишите их назначение.
- 3) Какие способы доработки цифрового материала из интернета вы знаете? На какие из них нужно брать разрешение у авторов?

2.2. Тема:

Базы данных

Одним из важнейших видов информационных ресурсов современного общества являются базы данных (БД). В настоящее время все крупные интернет-ресурсы используют БД для хранения информации, так как это позволяет упорядоченно хранить данные о группе объектов, обладающих одинаковым набором свойств.

Основные параметры БД – это ее физический и логический объемы. Физический объем выражается в байтах, Кб и так далее; логический зависит от числа записей. Крупнейшие базы данных могут составлять десятки терабайт и десятки миллиардов записей.

Набор правил структурирования данных, связей между этими данными и допустимых операций над ними называют **моделью организации баз данных**.

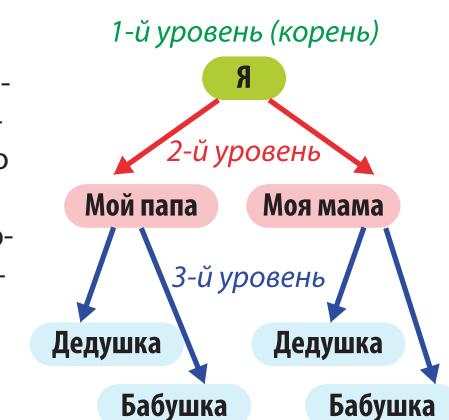
- 1 Различают три основные модели базы данных:
- 2 Иерархическая
- 3 Сетевая
- 4 Реляционная

Иерархическая модель представляет собой древовидную структуру, вершины которой состоят из записей. Например, структура каталогов, отражающая информацию о файлах, которые хранятся на компьютере. Эта модель имеет только одну вершину первого уровня, называемую корнем. Узел каждого уровня, кроме корня, связан только с одним узлом верхнего уровня и с несколькими узлами нижнего уровня. По другому такая связь называется «один-ко-многим».



ЗАПОМНИ

База данных (далее БД) – это набор сведений, логически систематизированных с целью их эффективного поиска и обработки в вычислительной системе.





Сетевая модель, кроме вертикальной связи, может иметь и горизонтальные связи, то есть каждый элемент одного уровня может быть связан с любым количеством элементов другого уровня.

Примером может служить база данных, в которой хранятся сведения об увлечениях одноклассников: один ученик может иметь много увлечений, а одно увлечение может быть у многих учеников.

Такая связь называется «многие-ко-многим».

Реляционная модель (англ. *relation* – *отношение*) – это совокупность таблиц с установленными связями между ними. В реляционной базе данных могут использоваться все виды связей: «один-к-одному», «один-ко-многим», «многие-ко-многим».

Строки в реляционной БД называются записями, а столбцы – полями. Каждая таблица должна содержать ключевое поле (ID), которое позволяет однозначно идентифицировать каждую запись в таблице. Поэтому большинство современных систем управления базами данных (далее СУБД) ориентируются на реляционные БД.

Наиболее популярные СУБД: MySQL, Oracle, PostgreSQL, SQLite, Firebird, DB2, Microsoft Access, Microsoft SQL Server.

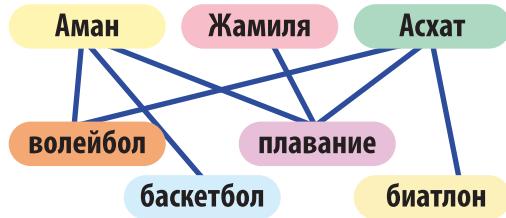


Система управления базами данных OpenOffice.org Base

Для обработки сложных взаимосвязанных данных, содержащих информацию разных категорий, используются системы управления базами данных (СУБД).

Рассмотрим принципы работы СУБД на примере OpenOffice.org Base. К примеру, нам нужно создать онлайн галерею картин. Для этого нужно создать базу данных, где хранится вся актуальная информация по каждой картине:

- а) автор картины;
- б) год, когда она была нарисована;
- в) жанр (портрет, натюрморт или пейзаж);
- г) название картины.



Теперь, зайдя на сайт, посетитель онлайн галереи может выбрать для просмотра картины только конкретного автора или только портретный жанр.

В левой части основного окна программы расположены четыре основные кнопки — *таблицы, запросы, формы и отчеты*, которые позволяют создавать:

- **Таблицы** – для хранения информации.
- **Запросы** – для отбора информации по заданным критериям.
- **Формы** – для быстрого и комфорtnого ввода данных в таблицы.
- **Отчеты** – для вывода информации в удобном для пользователя виде, в том числе и для печати.

Создание базы данных

При запуске программы автоматически запускается Мастер создания и открытия готовых БД. Для создания новой БД необходимо:

- 1 Выбрать пункт «**Создать новую базу данных**» и нажать «Далее».
- 2 Второй шаг имеет два вопроса с двумя ответами для каждого. Ответом по умолчанию на первый вопрос является «Да, зарегистрировать базу данных», и ответом на второй вопрос будет: «Открыть базу для редактирования». Нажать кнопку «Готово».
- 3 Далее сохранить базу данных под каким-либо именем, например, *Gallery*.

Создание таблицы БД

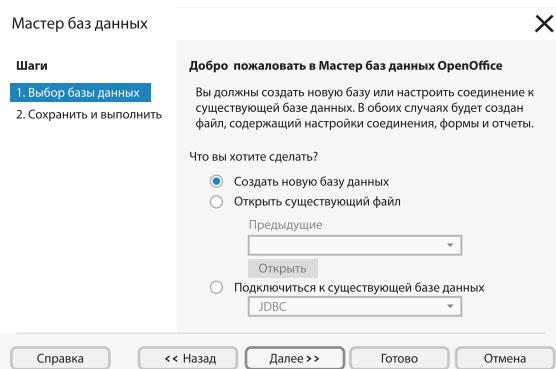
Для создания таблицы БД в окне «База данных» необходимо выбрать значок «Таблицы» и выбрать один из способов создания таблицы:

- Создать таблицу в режиме дизайна, то есть самостоятельно задать имена полей, типов данных, и т.д.



ЗАПОМНИ

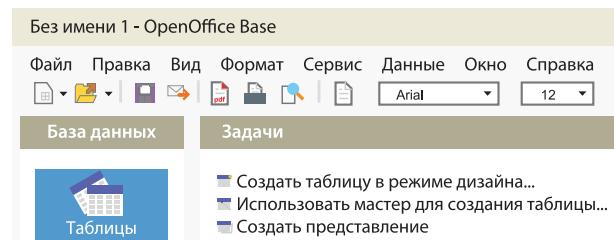
Строки в БД называются записями, а столбцы называются полями. Каждая таблица должна содержать ключевое поле (ID), которое позволяет однозначно идентифицировать каждую запись в таблице.





- Использовать мастер для создания таблиц, то есть выбрать из готовых примеров таблицу с полями.

Для создания БД картин выберем «Создать таблицу в режиме дизайна», в открывшемся окне введем «Названия поля» и «Тип поля»:



Название поля	Тип поля
автор	Текст [VARCHAR]
название	Текст [VARCHAR]
жанр	Текст [VARCHAR]
год	Дата [DATE]

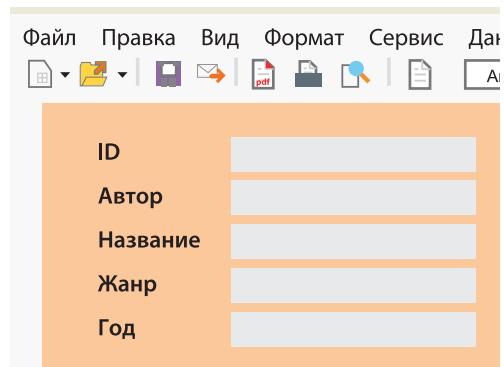
Поля таблиц могут иметь различные типы данных, например:

- текстовый
- логический
- числовой
- дата / время

При первом закрытии и сохранении файла программа спросит «Создать первичный ключ?». Необходимо выбрать «Да», при этом в вашей таблице появится еще одно поле с названием «ID».

Создание формы

Теперь создадим интерфейс пользователя, то есть форму, через которую можно вводить данные в таблицу БД. Для этого, нажав на кнопку Формы, выберите пункт «Мастер формы». Мастер проведет вас через довольно простые шаги при создании формы на основе уже созданной вами таблицы. Здесь же вы сможете выбрать внешний вид формы.



Построение запросов

На основе таблиц баз данных можно создавать запросы, то есть отфильтровывать необходимые вам данные. Открыв запрос, вы увидите текущие данные в виде таблицы, которую вы задали в параметрах запроса.

Так же как с таблицами и формами, запросы можно создавать в режиме «Мастер запросов» или в режиме «Дизайн запросов».

Чтобы определить запрос, необходимо указать имена полей базы данных, которые требуется включить, а также условия отображения полей. В БД нашей галереи, например, можно отобрать картины конкретных авторов и только в портретном жанре.

Создание отчетов

Отчет – это текстовый документ, отображающий данные вашей БД. Вы можете изменять размер, внешний вид и структуру отчета.

Отчеты могут показывать данные одной или нескольких таблиц и включать любые комбинации полей из каждой таблицы или запроса, на основе которых они создаются.

Для быстрого создания отчета можно использовать «Мастер создания отчетов». В нем нужно поэтапно:

- а) выбрать источник данных, таблицу или запрос, из которых будут браться данные;
- б) выбрать поля, которые будут отображаться;
- в) определить внешний вид отчета (расположение полей, тип отображения элементов и фон).

Теперь вы можете распечатать готовый отчет.

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Приведите примеры иерархической и сетевой баз данных. Создайте их схемы с помощью средств OpenOffice.
- 2) Создайте базу данных «Мои друзья» с использованием следующих типов полей: текстовое, числовое, дата/время, логическое.
- 3) Разработайте базу данных библиотеки, которая содержит следующую информацию: название книги, Ф.И.О. авторов, наименование издательства, год издания, количество страниц, количество иллюстраций, количество имеющихся экземпляров конкретной книги, количество учащихся, которым выдавалась конкретная книга. База данных должна включать таблицу, запрос и форму.

Глава

3



Программирование



3.1. Тема:

Сложные условия: and, or, not

В программировании важно уметь правильно выстраивать условия. Часто условия бывают сложными, т.е. состоят из нескольких составных условий, соединенных логическими операторами (связками) «И», «ИЛИ» и «НЕ». В Python они записываются английскими словами «and», «or» и «not».

Логический оператор **and** (логическое умножение)

Сложное условие, где составные условия соединены связкой **and**, возвращает **True**, если все выражения равны **True**. Если же хоть одно из выражений ложно, то всё условие является ложным:

```
x = 5
if x < 10 and x % 3 == 0:
    print('True')
else:
    print('False')
```

Ответом здесь будет **False**, потому что, согласно второй части условия, заданное число 5 не делится на 3 без остатка. Если бы мы записали выражение так: $x \% 3 == 0$ **and** $x < 10$, то оно также вернуло бы **False**. Однако второе сравнение $x < 10$ не выполнялось бы интерпретатором, так как его незачем выполнять. Ведь первое выражение ($x \% 3 == 0$) уже вернуло ложь, которая, в случае оператора **and**, превращает всё выражение в ложь.

Логический оператор **or** (логическое сложение)

Возвращает **True**, если хотя бы одно из выражений равно **True**:

```
x = 5
if x < 10 or x % 3 == 0:
    print('True')
else:
    print('False')
```

Ответом здесь будет **True**, потому что, согласно первой части условия, заданное число 5 меньше 10, хоть и не делится на 3 без остатка. Именно поэтому, если одно из выражений возвращает **True**, то второе выражение не оценивается, так как оператор **or** в любом случае возвратит **True**.

Логический оператор **not** (логическое отрицание)

Унарный оператор **not** превращает истину в ложь, а ложь в истину. Унарный он потому, что применяется к одному выражению, стоящему после него, а не справа и слева от него, как в случае бинарных **and** и **or**.

Вариант 1:

```
x = 8
print (not x < 15)
```

False

Вариант 2:

```
x = 8
print (not x > 15)
```

True

Если в одном выражении одновременно используется несколько или даже все логические операторы, приоритет операций следующий:

- 1) отношения (<, >, <=, >=, ==, !=)
- 2) not («НЕ»)
- 3) and («И»)
- 4) or («ИЛИ»)

Для изменения порядка действий используют круглые скобки. Рассмотрим пример изменения порядка вычислений в случаях, когда возникают скобки:

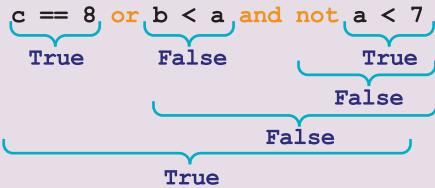
Вариант 1:

```
a=4
b=6
c=8
result = c==8 or b<a and not a < 7
print (result)
```

Результат:

True

Выясним почему:



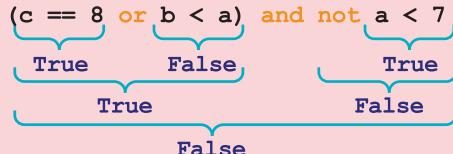
Вариант 2:

```
a=4
b=6
c=8
result= (c==8 or b<a) and not a < 7
print (result)
```

Результат:

False

Выясним почему:



ВОПРОСЫ И ЗАДАНИЯ:

- 1) С помощью оператора **and** составьте два сложных логических выражения, одно из которых дает истину, другое – ложь.
- 2) Аналогично выполните задачу 1, но уже с оператором **or**.

**3.2. Тема:**

Списки, кортежи и словари

Часто нам нужно держать много однообразных данных в одном файле, например, список учеников школы или номера телефонов в справочнике. В Python такие наборы данных можно организовывать в **списки, кортежи и словари**.

Список (list) – это структура, состоящая из элементов, расположенных в определенном порядке. Каждому элементу соответствует номер (или индекс), по которому к нему можно обратиться. Для создания списка в квадратных скобках ([])) через запятую перечисляются все его элементы. Например, создадим список членов своей семьи:

```
>>> myfamily = ['father', 'mother', 'sister', 'brother']
```

В данном случае наш список будет храниться в переменной myfamily. Когда список создан, можно написать программу для работы с этим списком. К примеру, напишем приветствие для каждого из членов семьи, используя цикл:

Программа на Python

```
myfamily = ['father', 'mother',  
           'sister', 'brother']  
for item in myfamily:  
    print('Hello', item)
```

Результат вывода на экран

```
Hello father  
Hello mother  
Hello sister  
Hello brother
```

Список может содержать разные типы объектов. В один и тот же список одновременно можно включать строки, числа, объекты других типов данных:

```
objects = [1, 2.6, 'Hello', True]
```

Списки можно складывать, тогда новый список будет содержать элементы из обоих списков:

```
x = [1, 2, 3, 4]  
y = [5, 6, 7, 8]  
z = x + y  
print (z)
```

Результат:

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

Со списками можно делать много разных операций:

x in A

Проверить, содержится ли элемент **x** в списке **A**.
Возвращает **True** или **False**

```
A = [1, 2, 3, 4, 5, 6, 7, 8]
print (2 in A)
Результат:
True
```

min(A)

Найти наименьший элемент в списке **A**.

```
A = [1, 2, 3, 4, 5, 6, 7, 8]
print (min(A))
Результат:
1
```

max(A)

Найти наибольший элемент в списке **A**.

```
A = [1, 2, 3, 4, 5, 6, 7, 8]
print (max(A))
Результат:
8
```

Кортеж (tuple), как и список, представляет собой последовательность элементов. Однако хранящиеся в нем элементы нельзя изменять, добавлять или удалять. Для создания кортежа используются круглые скобки, в которые помещаются его значения, разделенные запятыми:

```
user = ('Timur', 23, 1.10.1998)
print(user)
```

В кортежах удобно хранить свойства объектов, например, имя, возраст, дату рождения. Если вдруг кортеж состоит из одного элемента, то после единственного элемента кортежа необходимо поставить запятую:

```
user = ('Tom',)
```

Словари (dictionary) – это структура данных, в которой каждый элемент вместо индекса имеет уникальный ключ. Элементы словаря можно изменять. Для создания словаря используются фигурные скобки ({}):

```
dictionary = {ключ1:значение1, ключ2:значение2, ...}
```

Создадим словарь под именем **myschool**:

```
myschool = {'5 klass':'Anara, Kanat, Pavel', '6 klass':
'Chyngyz, Tina, Emil'}
```

В этом словаре в качестве ключей используются названия классов, а в качестве значений – имена тех, кто учится в этих классах.



В словарь можно добавить значение, пометив его новым ключом:

```
myschool['7 klass'] = 'Elena, Ainura, Dastan'  
print (myschool)
```

Результат:

```
{'5 klass': 'Anara, Kanat, Pavel', '6 klass': 'Chyngyz,  
Tina, Emil', '7 klass':'Elena, Ainura, Dastan'}
```

Чтобы изменить значение элемента, нужно придать его ключу новое значение:

```
myschool = {'5 klass':'Anara, Kanat, Pavel', '6 klass':  
'Chyngyz, Tina, Emil'}  
myschool['6 klass'] = 'Matvei, Tina, Salima'  
print (myschool)
```

Результат:

```
{'5 klass':'Anara, Kanat, Pavel', '6 klass':'Matvei, Tina, Salima'}
```

Используя цикл **for**, можно вывести на экран только ключи словаря:

```
for i in myschool:  
    print(i)
```

Результат:

```
5 klass  
6 klass
```

Или вывести только значения словаря:

```
for i in myschool:  
    print(myschool[i])
```

Результат:

```
Anara, Kanat, Pavel  
Chyngyz, Tina, Emil
```

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

Создайте словарь «Рыбы», а его элементы разделите на 3 вида: «речные», «озерные» и «морские рыбы». Выведите на экран сначала только ключи, а потом элементы словаря.

3.3. Тема:

Циклические алгоритмы

Доказано, что любой алгоритм может быть записан с помощью трех алгоритмических конструкций: циклов, условных операторов, линейных алгоритмов.



СМОТРИ ТАКЖЕ

Тема 3.4 7 класс

Циклы **while** и **for**

Как вы знаете, **цикл** – это многократное выполнение одинаковых действий.

В 7 классе мы уже начали изучать циклы **while** и **for**. Цикл **for** отличается от цикла **while** тем, что используется для повторения каких-то команд заранее известное количество раз. Цикл **while**, напротив, повторяет какое-то действие в тех случаях, когда мы не знаем, сколько повторений данной команды необходимо. При этом нам известно условие, до исполнения которого требуется повторять цикл.

Рассмотрим применение цикла **for** более подробно. Запись цикла **for** в Python осуществляется по схеме:



При старте цикла значение первого элемента диапазона будет присвоено переменной, с которой будет выполнено обозначенное действие. При втором проходе переменной будет присвоено значение второго элемента диапазона. И так далее, пока со всеми элементами в диапазоне не будет проведено заданное действие. Рассмотрим пример, в котором переменной **letter** каждый раз присваивается новый элемент из строки **Python**. Команда **print** выводит на экран каждую букву этой строки по одной:

```
for letter in 'Python':  
    print ('Буква:', letter)  
>>>  
Буква: P  
Буква: у  
Буква: т  
Буква: h  
Буква: о  
Буква: н
```



В примере ниже с каждым новым проходом значение переменной увеличивается на число из заданного диапазона:

```
f = 12
for i in range(1, 6):
    f = f + i
print (f)
>>>
27
```

Цикл «`for i in range(1,6)`» выполняется пять раз (6 – не включается). На каждом шаге цикла переменная `f` увеличивается на `i`. Первоначальное значение `f = 12`. В цикле значение изменяется:

- 1 проход: $f = 12 + 1 = 13$
- 2 проход: $f = 13 + 2 = 15$
- 3 проход: $f = 15 + 3 = 18$
- 4 проход: $f = 18 + 4 = 22$
- 5 проход: $f = 22 + 5 = 27$

Кратко можно записать так: $f = 12 + 1 + 2 + 3 + 4 + 5 = 27$

Аргументы для функции `range` передаются следующим образом:

- **`range(x)`** – перебирает все значения от 0 до `x`;
при этом число `x` не включается в диапазон;
- **`range(y, x)`** – перебирает все значения от `y` до `x`;
`x` также не включается в диапазон;
- **`range(y, x, s)`** – перебирает значения от `y` до `x`, с шагом `s`.

Например:

```
for i in range(0,15,3):
    print(i)
```

В данном случае цикл **`for`** переберет значения от 0 до 15 с шагом 3, в результате он выведет каждое третье число:

```
>>>
0
3
6
9
12
```

Также в качестве шага можно использовать отрицательные числа, тогда цикл будет перебирать значения в обратном направлении:

```
for i in range(100, 0, -10):  
    print(i)  
>>>  
100  
80  
60  
40  
20
```

В отличие от цикла **for**, который повторяет последовательность определенное количество раз, цикл **while** руководствуется не количеством, а логическим условием, потому вам не нужно точно знать, сколько раз необходимо выполнить код.

Код цикла **while** будет повторяться до тех пор, пока логическое условие истинно (True).

Задача 1. Давайте попробуем на основе этого цикла написать игру, в которой пользователь должен угадать число, загаданное компьютером. Напишем программу:

```
import random #загрузим библиотеку случайных чисел  
number = random.randint(1, 25) #выбор компьютером случайного числа  
choices = 0 #в переменную choices записываем количество попыток  
while choices < 5: #выполняет цикл до 5 попыток  
    print('Угадай число между 1 and 25: ') #предлагает пользователю ввести число  
    guess = input()  
    guess = int(guess) #число должно быть целым  
    choices = choices + 1 #с каждой попыткой счет увеличивается на 1  
    if guess == number: #если введенное число равно загаданному  
        break #остановить программу
```

Переменной **choices** присвоено значение 0, которое будет увеличиваться с каждой попыткой угадать число. Мы ограничим программу 5-ю попытками, чтобы программа не попала в бесконечный цикл.

Программа уже работает, но она не сообщает пользователю никаких результатов: пользователь не знает, угадал он число или нет. Результат выглядит так:

```
Угадай число между 1 and 25:  
5  
Угадай число между 1 and 25:
```



16

Угадай число между 1 and 25:

7

Угадай число между 1 and 25:

18

Угадай число между 1 and 25:

10

>>>

Для этого введем условные операторы, которые будут сообщать пользователю, что его число меньше или больше загаданного, и это поможет угадать число быстрее:

```
import random
number = random.randint(1, 25)
choices = 0
while choices < 5:
    print('Угадай число между 1 and 25:')
    guess = input()
    guess = int(guess)
    choices = choices + 1
    if guess < number: #если число пользователя меньше загаданного
        print('Мое число больше твоего')
    if guess > number: #если число пользователя больше загаданного
        print('Мое число меньше твоего')
    if guess == number: #если число пользователя равно загаданному
        break
if guess == number:
    print('Молодец! Ты угадал число с' + str(choices) + 'попытки!')
else:
    print('К сожалению, ты не угадал число. Я загадал' + str(number))
```

Если запустить программу, то вариант общения с пользователем будет такой:

Угадай число между 1 and 25:

6

Мое число больше твоего

Угадай число между 1 and 25:

17

Мое число меньше твоего

Угадай число между 1 and 25:

14

Мое число больше твоего
Угадай число между 1 and 25:

15

Молодец! Ты угадал число с 4 попытки!
>>>

Теперь программа помогает пользователю угадать число, дает ему подсказки. К примеру, если компьютер загадал число 15, а пользователь ввел 17, программа подскажет, что введенное число больше загаданного.

ВОПРОСЫ И ЗАДАНИЯ:

1) Запишите значение переменной *y*, полученное в результате работы следующей программы.

```
y = 5
for i in range(2, 6):
    y = y + 4 * i
print (y)
```

2) Дан бруск длиной 20 метров. Напишите программу, которая посчитает, какое минимальное целое количество отрезков длиной 1,5 м и 2 м получится из данного бруска.

3) По данному коду запишите в таблицу значения переменных на каждом шаге алгоритма:

$k=4$ $p=1040$ $m=2$

```
while p != m*m:
    k=k+1
    p=p-4
    m=m*2
```

```
print (k)
```

k	p	m	$m*m$

4) У героя Майнкрафта Алекса есть машина, которая выпускает по 4 минерала в минуту. На каждые 100 минералов можно построить новую машину, которая также выпускает по 4 минерала в минуту. Напишите программу, которая вычислит, сколько машин будет у Алекса через час.



3.4. Тема:

Вложенные условные операции и циклы

В 7 классе мы уже изучали, как работают условные операторы **if** и **else**, которые реализуют в программе две отдельные ветви выполнения. Однако алгоритм программы может предполагать выбор больше, чем из двух путей, например, из трех, четырех или даже больше.

Внутри условного оператора могут находиться любые операторы, в том числе и другие условные операторы. Например, у нас есть показания датчика температуры из печи, требуется определить, высокая она, низкая или в пределах нормы. Нормальной считается температура в диапазоне от 200 до 250 градусов Цельсия. В переменной **t** хранится температура. Один условный оператор не подойдет, потому что есть три возможных результата. Решение задачи можно записать так:

```
t = int(input('Введите температуру'))
if t > 250:
    print('Температура в печи очень высокая')
else:
    if 200 <= t <= 250:
        print('Температура в печи в пределах нормы')
    else:
        print('Температура в печи ниже нормы')
```

Условный оператор **if**, проверяющий равенство, находится внутри блока **else (иначе)**, поэтому он называется **вложенным** условным оператором. Как видно из этого примера, использование вложенных условных операторов позволяет выбрать один из **нескольких** вариантов. Если после **else** сразу следует еще один оператор **if**, можно использовать «каскадное» ветвление с ключевыми словами **elif** (сокращение от **else-if**). Рассмотрим задачу, в которой потребуется «каскадное» ветвление: требуется определить четверть координатной плоскости по заданным координатам **x** и **y**. В переменных **x** и **y** хранятся целочисленные значения координат, введенные с клавиатуры.

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
```

```
print('Первая четверть')
elif x > 0 and y < 0:
    print('Четвертая четверть')
elif y > 0:
    print('Вторая четверть')
else:
    print('Третья четверть')
```

В приведенной программе условия `if`, ..., `elif` проверяются по очереди и выполняется блок, соответствующий первому из истинных условий. Если все проверяемые условия ложны, то выполняется блок `else`, если он присутствует.

Вложенные циклы

В сложных задачах часто бывает так, что на каждом шаге цикла нужно выполнять обработку данных, которая также представляет собой циклический алгоритм. В этом случае получается конструкция «цикл в цикле» или «вложенный цикл».

Цикл называется **вложенным**, если он размещается внутри другого цикла. На первом проходе внешний цикл вызывает внутренний, который исполняется до своего завершения, после чего управление передается в тело внешнего цикла. На втором проходе внешний цикл опять вызывает внутренний. И так до тех пор, пока не завершится внешний цикл.

Задача 1. Выведем на экран таблицу умножения. Для этого во внешнем цикле надо перебрать числа от 1 до 9. Для каждого из них нужно перебрать во внутреннем цикле числа от 1 до 9.

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18

На одну синюю цифру приходится один ряд черных цифр до 9-ти. Синие 1 и 2 находятся во внешнем цикле, черные цифры во внутреннем.

При этом во внутреннем цикле нужно выполнить умножение переменных счетчиков внешнего и внутреннего цикла первого ряда.

Таким образом, на одно выполнение внешнего цикла произойдет девять выполнений внутреннего, и сформируется одна строка таблицы умножения. После каждой строки надо перейти на новую: это делается во внешнем цикле, после того как закончится выполняться внутренний.



Также для построения таблицы необходимо использовать форматированный вывод, т.е. задавать ширину столбцов (`\t`), иначе произойдет сдвиг, т.к. количество цифр в каждой строке различно.

Наш код будет выглядеть так:

```
for i in range(1,10): #первый множитель от 1 до 10
    for j in range(1,10): #второй множитель от 1 до 10
        print(i*j, end='\t')
    print()
```

Результат:

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

Задача 2. Составим программу нахождения всех простых чисел в интервале от 2 до 100.

Простое число – это число, которое делится без остатка только на 1 и на само на себя. Например, число 5 – простое, так как его можно нацело разделить только на 5 и 1, а число 6 – составное, так как помимо 6 и 1 делится на 2 и 3.

- Единица не является простым числом, поэтому в нашем примере диапазон начинается с цифры 2.
- Согласно условию, если число `n` не имеет делителей в диапазоне от 2 до `n-1`, то оно простое, а если хотя бы один делитель в этом интервале найден, то составное.
- Проверим делимость числа `n` на некоторое число `k`: если остаток равен нулю, то `n` делится на `k`.
 - Когда найден хотя бы один делитель, число уже заведомо составное, и искать другие делители в данной задаче не требуется. Для этого при `n % k == 0` выполняется **досрочный выход из цикла с помощью оператора `break`**.
 - Переменная `flag` показывает, есть ли хоть один делитель у числа.

Таким образом, программу можно записать так (здесь `n`, `k` – целочисленные переменные):

```
for n in range(2,101):
    flag = False
    for k in range(2,n):
        if n % k == 0:
            flag = True
            break
    if not flag:
        print (n)
```



ЗАПОМНИ

Чтобы цикл повторился `n` раз, последним числом диапазона должно быть `n+1`

Задача 3. Нарисуем на экране «прямоугольник» из двух видов символов. Края прямоугольника будут отрисованы символом «0», а внутренняя часть символом «1».

Пусть длина прямоугольника будет равна 10 символам, а ширина 7. Внешний цикл, перебирая строки, первой и последней цифрой должен поставить 0. Если строка первая или последняя (поскольку отсчет идет от 0, то это 0-я и 6-я строка), 0 выстраивается от начала и до конца. Во всех остальных случаях – ставим цифру 1. Запишем программу:

```
for i in range(7):                      #выводим строку 7 раз
    if i==0 or i==6:                      #если строка 1-ая или последняя
        for j in range(20):               #все 20 раз
            print('0',end=' ')
    else:                                #иначе
        print('0',end=' ')              #выводим 0
        for j in range(1,19):           #кроме 1-ого и 19-ого которые
            print('1',end=' ')          #выводим цифрой 1
        print('0',end=' ')
print()
```

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.
- 2) Даны ящики, которые вмещают 5 кг, 10 кг и 15 кг яблок. Необходимо выяснить, сколько ящиков разного размера понадобится для того, чтобы распределить 100 кг яблок.



3.5. Тема:

ФУНКЦИИ

Ранее вы уже использовали **встроенные** в интерпретатор функции:

- `print()` – выводит на печать данные, заключенные в круглые скобки;
- `str()` – преобразует данные к строковому типу;
- `int()` – преобразует данные к целому числу;
- `float()` – преобразует целые числа в дробный тип;
- `round()` – округляет число в большую по модулю сторону.

Кроме них мы можем создавать свои собственные функции для выполнения тех или иных задач. Для этого в Python предусмотрена возможность, когда некоторый повторяющийся алгоритм (или его фрагмент) оформляется в отдельную функцию.

Для этого новой функции необходимо присвоить имя и описать его алгоритм. В дальнейшем, при упоминании в программе имени функции, запускается соответствующий алгоритм со списком входных и выходных данных. После выполнения функции работа продолжается с той команды, которая непосредственно следует за вызовом функции.

Предположим, что в нескольких местах программы требуется выводить на экран сообщение об ошибке: «Ошибка в программе». Это можно сделать, например, так:

```
print ('Ошибка в программе')
```

Вставка этого оператора вывода везде, где нужно вывести сообщение об ошибке, приведет к загруженности памяти. Если потребуется поменять текст сообщения, то нужно будет искать эти операторы вывода по всей программе. Именно для таких случаев используются функции – вспомогательные алгоритмы, к которым можно обратиться с другого места программы. Запишем функцию `error`:

```
def error():  
    print ('Ошибка в программе')  
n = int (input())  
if n < 0:  
    error()
```

Мы ввели новую функцию `error`.

Имя функции начинается с ключевого слова **def** (от англ. *define* – определить), после которого задается уникальное название функции (например, **def sum**). После имени функции ставятся скобки, в которых можно передавать параметры функции и двоеточие. Тело функции записывается с отступом. Для того чтобы функция заработала в другом месте программы, необходимо ее вызвать по имени (не забыв скобки). Например, **error()**.

Использование функций сокращает код, если какие-то операции выполняются несколько раз в разных местах программы. Иногда большую программу разбивают на несколько функций для удобства и упрощения, оформляя в виде функций отдельные этапы сложного алгоритма. Такой подход делает всю программу более понятной.

Функции и их аргументы

Функциям можно передавать аргументы – дополнительные данные для изменения выполняемых действий.

Предположим, что требуется многократно вывести на экран символ, например, чтобы нарисовать таблицу или разделитель. Программу, решающую эту задачу для переменной *n*, можно записать так:

```
n = 125 #количество раз
s = '-' #символ
while n > 0:
    print (s, end = ' ')
    n -= 1;
```

Обратим внимание на вызов функции **print** с именованным аргументом **end** – завершающий символ (по умолчанию – символ «новая строка»).



ЗАПОМНИ

Имена функций должны состоять из строчных букв, а слова разделяться символами подчеркивания – это делает код более удобным для чтения (*snake case*).

Вспомогательный алгоритм цикла вывода повторяющегося символа можно оформить в виде функции. Этой функции нужно передать *аргументы* – символ и число, сколько раз его повторить. Программа получается такая:

```
def print_char(s, n): #имя функции с аргументами
    k = n
    while k > 0:
        print (s, end = ' ')
        k -= 1
print_char ('-', 10) #аргументы
```



Основная программа содержит всего одну команду – вызов функции `print_char`. В скобках указаны **аргументы функции**, указывающие, что символ тире («-») нужно вывести 10 раз.

Глобальные и локальные переменные

Во многих случаях функции используют для обработки данных. Эти данные могут быть глобальными либо локальными. **Локальные переменные** передаются в функцию через аргументы, указанные в круглых скобках после имени функции. Локальные переменные находятся в «зоне видимости» только этой функции и недоступны для всей остальной программы. **Глобальные переменные** доступны во всей программе. К ним можно обращаться по имени и получить связанное с ними значение.

Задача 1. Рассмотрим типы переменных на примере данной программы:

```
def rectangle():
    a = float(input('Ширина: '))
    b = float(input('Высота: '))
    s = a*b
    print('Площадь: ', s)
def triangle():
    a = float(input('Основание: '))
    h = float(input('Высота: '))
    s = 0.5*a*h
    print('Площадь: ', s)
figure = input('1-прямоугольник, 2-треугольник: ')
if figure == '1':
    rectangle()
elif figure == '2':
    triangle()
```

В этой задаче 5 переменных, где глобальная только `figure`. Переменные `a` и `b` из функции `rectangle()`, и `a` и `h` из функции `triangle()` – локальные. При этом локальные переменные в разных функциях являются разными переменными.

Возврат значений из функции

Каждая функция выдает определенный результат. Если даже вы не указываете на возврат значения в качестве результата, она, тем не менее, выдаст результат `None` (ничего).

Для того чтобы указать, чему равно значение функции, используют оператор **return** (англ. вернуть), после которого записывают значение-результат.

Результатом может быть число, символ, символьная строка или любой другой объект.

Задача 2. Составим функцию, которая вычисляет сумму всех цифр числа (н-р, для числа 147 нужно сложить цифры: $1+4+7=12$). Сложение цифр начинаем с последней, в нашем примере это цифра 7.

- 1 Чтобы получить последнюю цифру числа, нужно взять остаток от деления числа на 10 ($147 \% 10 = 7$).
- 2 Полученный остаток добавляем к «сумме», которая изначально равна нулю ($sum=0$). Теперь сумма равна 7.
- 3 Затем мы «отсекаем» последнюю цифру числа 147, используя оператор целочисленного деления ($147 // 10 = 14$).
- 4 Поскольку цифра $14 > 0$, то мы возвращаемся в начало цикла. Цикл продолжается до тех пор, пока значение **n** не становится равно нулю.
- 5 В нашем примере мы снова отсекаем цифру 4 ($14 \% 10 = 4$) и добавляем ее к сумме ($4+7=11$).
- 6 Отделяем ее от всего числа ($14 // 10 = 1$).
- 7 Последнее однозначное число прибавляем к сумме ($11+1$).

Результат: 12

Таким образом, наша программа будет выглядеть так:

```
n = int(input('Введите число: '))
def digits_sum (n):
    total = 0
    while n > 0:
        total += n%10
        n = n // 10
    return total
#основная программа
print (digits_sum(n))
```

Задача 3. Напишем программу, которая определяет: делится ли сумма цифр заданного числа на 3. Причем, если в полученной сумме больше одной цифры, то необходимо повторить операцию, пока сумма не будет состоять из единственной цифры.



Например, сумма цифр числа 123456789 равняется 45. Число двузначное – опять находим сумму цифр: $4 + 5 = 9$. Получили девятку, которая делится на три ($9\%3==0$). Значит, исходное число 123456789 делится на три. Исходя из этого алгоритма, получим следующую программу:

```
def sum(n):
    sum = 0
    while n>0:
        sum += n % 10
        n = n // 10
    return sum

#основная программа
k = int(input('Введите число: '))
while k > 9: #пока сумма цифр не будет одной цифрой
    k = sum(k)          #вызываем функцию
if k%3==0:
    print ('Число делится на 3')
else:
    print ('Число не делится на 3')
```

Функцию можно вызывать не только из основной программы, но и из другой функции. Например, функция, которая находит среднее из трех различных чисел (то есть число, заключенное между двумя остальными), может быть определена так:

```
def middle ( a, b, c ):
    mi = min ( a, b, c )
    ma = max ( a, b, c )
    return a + b + c - mi - ma
```

Она использует встроенные функции `min` и `max`. Идея решения состоит в том, что если из суммы трех чисел вычесть минимальное и максимальное, то получится как раз третье число.

Функция может возвращать несколько значений. Например, возврат сразу и частного, и остатка от деления двух чисел можно написать так:

```
def divmod ( x, y ):
    d = x // y
    m = x % y
    return d, m
```

Результат функции можно записать в две различные переменные:

```
a, b = divmod ( 7, 3 )
print ( a, b ) # 2 1
```

Если указать только одну переменную, мы получим кортеж – набор элементов, который заключается в круглые скобки:

```
q = divmod ( 7, 3 )
print ( q ) # (2, 1)
```

В случаях, когда необходимо создать функцию, которая нужна в программе только один раз и выполняет несложную операцию с несколькими аргументами, используют **lambda-функции**. Lambda-функция – это анонимная функция, то есть не имеет своего названия, как в случае с **def**. Запись функции начинается со слова **lambda**, затем через пробел указываются аргументы функции и сразу после двоеточия указывается операция, результат которой будет возвращен функцией. Создадим такую функцию на примере умножения двух чисел:

```
multiple = lambda x, y: x * y #lambda-функция с 2-мя аргументами
print (multiple (2, 5)) #результат 10
```

В примере выше мы присвоили lambda-функцию к переменной **multiple**. Хотя такую функцию можно также записать одной строкой, не присваивая к переменной:

```
print ((lambda x, y: x * y) (2, 6))
```

Одно из главных умений в программировании – это не писать один и тот же код несколько раз. Как только это происходит, нужно понимать, что этот повторяющийся кусок кода должен идти в функцию.

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Напишите функцию, которая выводит на экран три переданные ей числа в порядке возрастания.
- 2) Напишите функцию, которая находит наибольший общий делитель двух натуральных чисел.
- 3) Напишите функцию, возвращающую знак числа, где аргументом является целое число. Возвращаемый результат 0 – если аргумент равен нулю. -1 – если число отрицательное, 1 – если число положительное.

**3.6. Тема:**

Массивы

Основное предназначение современных компьютеров – обработка большого количества данных. При этом надо как-то обращаться к каждой из тысяч (или даже миллионов) ячеек с данными. В этой ситуации имя дают не ячейке, а группе ячеек, в которой каждая ячейка имеет собственный номер. Такая область памяти называется массивом (или таблицей).

Массив – это группа переменных, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка в массиве имеет уникальный номер (индекс).

Одномерные массивы в Python представляют собой список элементов. Поэтому для работы с массивами используются списки (тип данных **list**).

Список в Python – это набор элементов, каждый из которых имеет свой номер (индекс). Нумерация всегда начинается с нуля, второй по счету элемент имеет номер 1 и т.д.

Создавать списки можно разными способами. Самый простой способ сделать это – перечислить элементы списка через запятую в квадратных скобках:

```
a = [1, 3, 4, 23, 5]
```

Список – это динамическая структура, его размер можно изменять во время выполнения программы (удалять и добавлять элементы).

Списки можно «складывать» с помощью знака «+», например, показанный выше список можно было построить так:

```
a = [1, 3] + [4, 23] + [5]
```

Сложение одинаковых списков заменяется умножением «*». Вот так создается список из 10 элементов, заполненный нулями:

```
a = [0]*10
```

В более сложных случаях используют генераторы списков – выражения, напоминающие цикл, с помощью которых заполняются элементы вновь созданного списка:

```
a = [i for i in range(10)]
```

**СМОТРИ ТАКЖЕ**

Тема 3.2 8 класс

Списки, кортежи и
словари

Как вы знаете, цикл `for i in range(10)` перебирает все значения `i` от 0 до 9. Выражение перед словом `for` (в данном случае – `i`) – это то, что записывается в очередной элемент списка для каждого `i`. В приведенном примере список заполняется значениями, которые последовательно принимает переменная `i`, то есть получим такой список:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

То же самое можно получить, если использовать функцию `list` для того, чтобы создать список из данных, которые получаются с помощью функции `range`:

```
a = list(range(10))
```

Для заполнения списка квадратами этих чисел можно использовать такой генератор:

```
a = [i*i for i in range(10)] #[0,1,4,9,16,25,36,49,64,81]
```

В конце записи генератора можно добавить условие отбора. В этом случае в список включаются лишь те из элементов, перебираемых в цикле, которые удовлетворяют этому условию.

Например, следующий генератор составляет список из всех четных чисел в диапазоне от 0 до 9:

```
a = [i for i in range(10) if i%2 ==0] #[0,2,4,6,8]
```

Часто в тестовых и учебных программах списки заполняют случайными числами. Это тоже можно сделать с помощью генератора:

```
from random import randint
a = [randint(20,100) for x in range(10)]
```

Здесь создается список из 10 элементов и заполняется случайными числами из отрезка [20,100]. Для этого используется функция `randint`, которая импортируется из модуля `random`.

Длина списка (количество элементов в нем) определяется с помощью функции `len`. К примеру:

```
a = [1, 3, 4, 23, 5]
n = len(a) #5
```



Далее во всех примерах мы будем считать, что в программе создан список **a**, состоящий из **n** элементов (целых чисел). Переменная **i** будет обозначать индекс элемента списка.

Для создания списка из 3-х элементов с использованием цикла запишем следующую программу:

Программа на Python

```
n = 3
a = [0]*3
for i in range(n):
    print('a[', i, ']=' , sep=' ', end="")
    a[i] = int(input())
```

Результат вывода на экран

```
a[0] = 1
a[1] = 2
a[2] = 3
```

Создать список из **n** элементов и ввести их значения можно также с помощью генератора списка:

```
a = [int(input()) for i in range(n)]
```

Здесь пользователь должен ввести **n**-количество элементов, которые преобразуются в целое число с помощью функции **int**, и это число добавляется к списку.

Возможен еще один вариант ввода, когда все элементы списка вводятся в одной строке. В этом случае строку, полученную от функции **input**, нужно «расщепить» на части с помощью метода **split**:

```
data = input()
s = data.split()
print (s)
```

Для вывода списка нужно использовать операцию **print**. Для вывода каждого элемента списка с указанием его индекса в квадратных скобках и по одному элементу в строке используем следующую программу:

Программа на Python

```
a = [1, 2, 3, 4, 5];
for i in range(0, len(a)):
    print('a[', i, ']=' , a[i],
          sep='')
```

Результат вывода на экран

```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
```

Перебор элементов

Перебор элементов списка состоит в том, что мы в цикле просматриваем все элементы списка и, если нужно, выполняем с каждым из них некоторую операцию. Переменная цикла изменяется от 0 до $n-1$, где n – количество элементов списка:

```
for i in range(0,n):  
    a[i] += 1;
```

в этом примере все элементы списка «`a`» увеличиваются на 1.

Если сам список изменять не нужно, для перебора его элементов удобнее всего использовать такой цикл:

```
a = [1,2,3,4,5]  
for x in a:  
    print(x)
```

Здесь вместо `print(x)` можно добавлять любые другие операторы, работающие с копией элемента, записанной в переменную «`x`». Обратите внимание, что изменение переменной «`x`» в теле цикла приведет к ошибке.

Во многих задачах требуется найти некоторые элементы списка, удовлетворяющие заданному условию, и как-то их обработать. Простейшая из таких задач – подсчет нужных элементов. Для решения этой задачи нужно ввести переменную-счетчик, начальное значение которой равно нулю. Далее в цикле просматриваем все элементы списка. Если для очередного элемента выполняется заданное условие, то увеличиваем счетчик на 1.

Предположим, что в списке «`a`» записаны данные о росте детей в классе. Найдем количество учеников, рост которых больше 120 см, но меньше 150 см. В следующей программе используется переменная-счетчик `count`:

```
count = 0  
for x in a:  
    if (120 < x < 150):  
        count += 1
```

Теперь более сложная задача: требуется найти средний рост этих детей. Для этого нужно дополнительно в отдельной переменной складывать все нужные значения, а после завершения цикла разделить эту сумму на общее количество значений. Начальное значение переменной `sum`, в которой накапливается сумма, тоже должно быть равно нулю:



```
count = 0
sum = 0
for x in a:
    if 120 < x < 150:
        count += 1
        sum += x
print (sum / count)
```

Суммирование элементов списка – это очень распространенная операция, поэтому для суммирования элементов в Python существует встроенная функция `sum`:

```
a = [1,2,3,4,5]
print (sum(a))
```

С ее помощью можно решить предыдущую задачу более аккуратно: сначала выделить в дополнительный список все нужные элементы, а затем поделить их сумму на количество (длину списка).

Для построения нового списка будем использовать условный оператор:

```
b = [x for x in a if 120 < x < 150]
print (sum(b)/len(b))
```

Условие отбора передало из списка «`a`» в новый список «`b`» только те элементы, которые удовлетворяют условию. А для вывода среднего роста детей в классе остается разделить сумму элементов нового списка на их количество.

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) В списке хранятся оценки учеников. Напишите программу, которая выводит номера элементов списка, равных введенной с клавиатуры оценке.
- 2) На физкультуре рост учеников записали в список. Найдите в этом списке ученика с самым высоким и низким ростом.

Тема 3.7.

Строки и операции с ними

Изначально компьютеры использовались как вычислительные машины, тогда как сейчас их главной задачей становится обработка текстовой информации. Основной тип данных для работы с текстом в языке Python – это строки (тип **str**) от англ. *string*.

Строка – это последовательность символов, заключенных в одинарные или двойные кавычки: 'Это строка' = "Это строка"

В отличие от списков (массивов), строки не относятся к структурам данных. При этом строки можно рассматривать как упорядоченную последовательность элементов, с которыми можно работать так же, как и с элементами в списках.

```
>>> s = 'Это строка'  
>>> s[0] #возвращает элементы под указанным индексом  
'Э'  
>>> s[5:] #возвращает элементы с 5-индекса и до последнего  
'трука'
```

Однако строки в Python неизменяемы. То есть нельзя заменить какой-то отдельный элемент строки на другой. В этом случае программа выдаст ошибку.

Можно составить из символов существующей строки новую строку, и при этом внести нужные изменения. Приведем полную программу, которая вводит строку с клавиатуры, заменяет в ней все буквы «а» на буквы «б» и выводит полученную строку на экран.

```
s = input('Введите строку: ')  
s1 = ''  
for c in s: #перебирает все символы в строке s  
    if c == 'а': #если значение переменной совпадает с «а»  
        c = 'б' #то заменяем ее на букву «б»  
    s1 = s1 + c  
print (s1)
```

Однако этот способ работает очень медленно. В практических задачах, где требуется замена символов, лучше использовать встроенный метод **replace**.



Кроме этого часто требуется строку текста обработать: получить часть этой строки (подстроку), объединить две строки в одну, либо удалить часть строки. Например, для **объединения** (сцепления) строк используется оператор '+'. Эта операция называется конкатенация:

```
s1 = 'Добрый'
s2 = 'день'
s = s1 + ' ' + s2 + '!'
print (s)
>>>
```

Добрый день!



ВСПОМНИ

Длина строки определяется с помощью свойства строки `len` (англ. `length` – длина). Для этого в переменную `n` записывается длина строки `s`, которая выдает количество знаков вместе с пробелами (целое число):

`n = len(s)`

Использование срезов для обработки строк

В языке Python часто используются срезы (англ. `slicing`), которые обозначают определенный участок строки для обработки. Запись среза выглядит так: `[X:Y]`. X – начало среза, а Y – окончание. Символ с номером Y в срезе не входит. По умолчанию первый индекс равен 0, а второй – длине строки.

Например, `s[3:8]` означает символы строки `s` с 3-го по 7-й (то есть до 8-го, не включая его).

Значение

Для выделения части строки (подстроки)

Примеры для строки `s = 'Понедельник'`

```
s1 = s[2:8]
#в строку s1 будет записано значение с
#3-его элемента по 8-й включительно
>>> недель
```

Для удаления части строки

```
s1 = s[:3] + s[9:]
#вырезаем с начала 3-элемента, и с 9-ого
#элемента до конца и складываем их в
#строке s1
>>> Поник
```

Вставить новый фрагмент внутри строки

```
s1 = s[:3] + 'ABC' + s[3:]
#после 3-его элемента вставляем ABC
>>> ПонABCедельник
```

Реверс строки (развернуть её наоборот):

```
s1 = s[::-1]
>>> кинъледеноН
```

Выбрать элементы через заданный шаг

```
s1 = s[::2]
#возвращает каждый второй символ
>>> Пндлнк
```

Методы строк

В Python есть множество встроенных методов для работы со строками. Рассмотрим наиболее интересные из них.

1 Методы **upper** и **lower** позволяют перевести строку соответственно в верхний и нижний регистр, метод **title** переводит только первые буквы в верхний регистр, а все остальные в нижний:

```
s = 'aAbB cC'  
s1 = s.upper() #'AABB CC'  
s2 = s.lower() #'aabb cc'  
s2 = s.title() #'Aabb Cc'
```

2 Метод **split** позволяет разбить строку по пробелам. В результате получается список из слов. Если пользователь вводит в одной строке ряд слов или чисел, каждое из которых должно в программе обрабатываться отдельно (как в списках), то без **split()** не обойтись:

```
s = 'Понедельник Вторник Среда Четверг Пятница'  
s1 = s.split () #['Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница']
```

3 Метод **join**, наоборот, формирует из списка строку. Для этого впереди ставится строка-разделитель, а в скобках передается список:

```
s = ['Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница']  
s1 = '-'.join (s) #Понедельник-Вторник-Среда-Четверг-Пятница
```

4 Метод **find** работает с подстроками. Он ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1.

```
s = 'Понедельник Вторник Среда Четверг Пятница'  
s1 = s.find ('Среда') #ответ: 20, индекс буквы «С», первого элемента в подстроке
```

Также этим методом можно найти номер индекса элемента строки в заданном срезе. Для того чтобы указать срез, нужно указать его начало и конец цифрами через запятую. Если вторая цифра не указана, то поиск ведется до конца строки:

```
s2 = s.find ('н', 4) #8, индекс первого «н», в отрезке с 4-ого индекса и до конца
```



`s1 = s.find ('н', 0, 9) #2, индекс первого «н», в отрезке от начала и 9-ого индекса`

5 Метод `replace` заменяет одну подстроку на другую. При этом исходная строка не меняется, а модифицируется новая строка, которая присваивается новой переменной `s1`:

```
s = 'Понедельник Вторник Среда Четверг Пятница'  
s1 = s.replace ('н', 'Н') #ПоНедельник Вторник Среда Четверг  
Пятница
```

Иногда нам нужно разделить строку, чтобы ее часть выводилась с новой строки, тогда мы используем знак `\n`:

```
print ('Понедельник \n Вторник \n Среда \n Четверг') #все  
 слова выводятся в столбик
```

Если новую строку нужно вывести с отступом, тогда используем знак `\t`:

```
print ('Понедельник \n\t Вторник \n\t Среда') #все слова выве-  
дются в столбик, каждая новая строка напечатается с отступом
```

Рассмотрим пример обработки строк с использованием изученных выше команд.

Задача 1. С клавиатуры вводится строка, содержащая имя, отчество и фамилию человека, например:

Айтматов Чынгыз Торекулович

Каждые два слова разделены одним пробелом, в начале строки пробелов нет. В результате обработки должна получиться новая строка, содержащая фамилию и инициалы:

Айтматов Ч.Т.

Решение:

1 Введем строку с клавиатуры:

```
s = input('Введите фамилию, имя и отчество: ')
```

2 Разобьем введенную строку на отдельные слова, которые разделены пробелами. Для этого используем метод `split`. В этом списке будут три элемента: `fio[0]` – фамилия, `fio[1]` – имя и `fio[2]` – отчество:

```
fio = s.split()
```

3 Соберем фамилию с инициалами:

```
fioshort = fio[0] + ' ' + fio[1][0] + '.' + fio[2][0] + '.'
```

Полная программа будет выглядеть так:

```
s = input('Введите фамилию, имя и отчество: ')
fio = s.split()
fioshort = fio[0] + ' ' + fio[1][0] + '.' + fio[2][0] + '.'
print (fioshort)
```

Сравнение и сортировка строк

Мы пользуемся сортировкой по алфавиту для того, чтобы быстрее найти слово. Если бы в словарях слова стояли не по алфавиту, мы бы тратили кучу времени на поиск одного слова. Каков же принцип сортировки строк в Python?

Оказывается, как и числа, буквы имеют свой вес. Буква, стоящая раньше в алфавите – легче, то есть «а» легче, чем «б», поэтому при сортировке она выходит первой. Из этого следует, что строки, как и числа, можно сравнивать.

При сравнении слов сначала сравниваются первые буквы, если они отличаются, то определяется результат сравнения. Дальнейшие буквы уже не сравниваются. Если первые буквы равны, то сравниваются следующие два элемента, и так до конца. Например, слово «паровоз» будет «меньше», чем слово «пароход»: они отличаются в пятой букве и «в» < «х».

Если у вас закончились символы для проверки, то более короткая строка меньше длинной, поэтому «пар» < «парк». Заглавные буквы легче строчных, потому что стоят раньше в алфавите. Поэтому «А» < «а», «П» < «а».

Но откуда компьютер «знает», что такое «алфавитный порядок»? Оказывается, при сравнении строк используются коды символов ASCII и Unicode, где каждый символ имеет свой порядковый номер.

Возьмем пару «ПАР» и «Пар». Первый символ в обоих словах одинаков, а второй отличается – в первом слове буква заглавная, а во втором – такая же, но строчная. Таким образом, получается, что:

«ПАР» < «Пар» < «пар»



А как же с другими символами (цифрами, латинскими буквами)? Цифры стоят в кодовой таблице по порядку, причем раньше, чем латинские буквы; латинские буквы – раньше, чем русские; заглавные буквы (русские и латинские) – раньше, чем соответствующие строчные. Поэтому

«5STEAM» < «STEAM» < «Steam» < «steam» < «ПАР» < «Пар» < «пар»

Например, для того чтобы отсортировать буквы внутри слова, программу можно записать так:

```
s = 'Понедельник'  
s1 = ''.join (sorted (s))  
print (s1)  
>>> Пдееиклиноъ
```

Задача 1. Необходимо ввести с клавиатуры несколько слов (например, фамилий) и вывести их на экран в алфавитном порядке.

Для решения этой задачи удобно перезаписать введенные через запятую фамилии в список, а затем отсортировать с помощью метода sorted:

```
s = input('Введите фамилии: ') #Абакиров Муканова Бебинов  
Семенова Запруда  
s1 = s.split() #формирует список из подстрок ['Абакиров',  
'Муканова', 'Бебинов', 'Семенова', 'Запруда']  
s2 = ''.join (sorted (s1))  
print (s2)  
>>> Абакиров Бебинов Запруда Муканова Семенова
```

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Напишите программу, которая запрашивает ввести несколько слов с клавиатуры, а затем определяет длину самого короткого слова в строке.
- 2) Строковый метод isdigit() проверяет, состоит ли строка только из цифр. Напишите программу, которая запрашивает с ввода два целых числа и выводит их сумму. В случае некорректного ввода программа не должна завершаться с ошибкой, а должна продолжать запрашивать числа. Обработчик исключений try-except использовать нельзя.

Тема 3.8.

Форматирование строк

Форматирование строк в Python означает подстановку в какое-либо место шаблона другого текста. Подстановка происходит, что называется, «на лету». Например, с использованием форматирования можно создавать такие пригласительные билеты через готовый шаблон:

Уважаемый (ая) Алина!
Приглашаем Вас на День открытых дверей.
Дата события: 1 мая
С уважением, Тимур.

Одна подстановка запишется так:

```
print ('Уважаемый (ая), {}'.format ('Алина'))
```

То есть, после текста шаблона вставляются пустые фигурные скобки и далее `.format()` с указанием в скобках значения, которое нужно вставить. При исполнении программы текст-подстановка вставляется на место фигурных скобок.

При множественных вставках в фигурные скобки вставляются индексы слов, которые идут в кортеже после `.format()`. Таким образом, запись шаблона для нашего приглашения будет выглядеть так:

```
print ('Уважаемый (ая), {0}! \nПриглашаем Вас на {1}. \nДата  
события: {2} \n С уважением, {3}'.format ('Алина', 'День  
открытых дверей', '1 мая', 'Тимур'))
```

Форматировать текст можно другим способом – без использования `.format()`. Этот способ считается менее правильным и устаревающим. Однако, если вы встретите в коде оператор `%` – то, возможно, речь идет именно о форматировании. Запишем наш шаблон приглашения с помощью оператора `%`:

```
print ('Уважаемый (ая), %s! \nПриглашаем Вас на %s. \nДата  
события: %d %s \n С уважением, %s.' % ('Алина', 'День открытых  
дверей', 1, 'мая', 'Тимур'))
```

```
>>>
```

Уважаемый (ая) Алина!

Приглашаем Вас на День открытых дверей.

Дата события: 1 мая

С уважением, Тимур.



Вы наверное заметили, что где-то мы записали %d, где-то %s? Они определяют, что мы используем в качестве подстановки:

- %s – подставляет строку;
- %d – подставляет целое число;
- %f – подставляет дробное число.

Преобразования число-строка и строка-число

В практических задачах часто нужно преобразовать число, записанное в виде цепочки символов, в числовое значение и наоборот. Для этого в языке Python есть стандартные функции:

int – преобразует строку в целое число

```
s = '123'  
N = int(s) #N = 123
```

float – преобразует строку в вещественное число (дробное)

```
s = '123.456'  
X = float(s) #X = 123.456
```

str – переводит целое или вещественное число в строку

```
N = 123  
s = str(N) #s = '123'  
X = 123.456  
s = str(X) #s = '123.456'
```

Если строку не удалось преобразовать в число (например, если в ней содержатся буквы), возникает ошибка и программа завершается.

Для дробных чисел (тип float) можно указать, сколько знаков в дробной части мы хотим вывести, например:

```
X = 34.8589578  
print('{:.2f}'.format(X)) #34.86  
print('{:.3f}'.format(X)) #34.859
```

Если в больших числах мы хотим использовать запятую в качестве разделителя разрядов, то записываем это так:

```
print('{:, .2f}'.format(10001.23554)) #10,001.24
```

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

Напишите шаблон по запросу у пользователя логина и пароля. Если логин или пароль будут введены с ошибкой, то необходимо вывести сообщение о том, что такой логин или пароль не найдены. Если логин и пароль совпадают, то вывести приветствие с указанием имени пользователя.

Тема 3.9.

Работа с графикой в Python

Рисование с помощью модуля **Turtle**

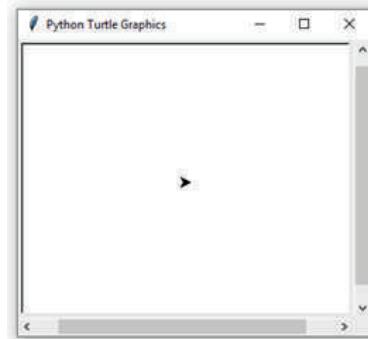
Для простого рисования в среде программирования Python используется модуль **Turtle** (черепашка), который подключается командой:

```
from turtle import* #загружает команды для работы с черепашкой
reset () #обнуляет позицию и включает перо
```

Запустив программу, вы увидите окно для графики с пером (черепашкой) по центру.

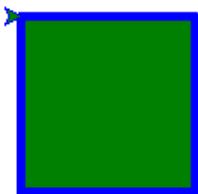
Теперь, задавая команды, мы можем строить рисунки на экране. Например, команда **forward** служит для перемещения черепашки вперед, где в скобках указывается количество шагов в пикселях. Для команд поворота **right** и **left** в скобках указывается количество градусов, на которые необходимо повернуть. Попробуем нарисовать квадрат:

```
from turtle import*
reset ()
forward (100) #движение вперед на 100
пикселей
right (90)    #поворот направо на 90°
forward (100)
right (90)
forward (100)
right (90)
forward (100)
```



Для сокращения кода используем цикл **for**, а также зададим дополнительные параметры для фигуры:

```
from turtle import*
width (5)           #ширина линии - 5 пикселей
color('blue','green') #первый - цвет линии, второй - цвет заливки
begin_fill() #начать следить за черепашкой для заполнения области
for i in range (4):
    forward (100)
    right (90)
end_fill() #заполнить цветом область, начиная с begin_fill()
>>>
```



Используя команду **up()**, вы можете поднять перо и с другого места начать рисовать другой рисунок (или напечатать текст). Для начала рисования нужно опустить перо командой **down()**.

Для того чтобы нарисовать круг используется команда **circle(r, n)**, где r – радиус круга, n – часть окружности, которую мы рисуем, в градусах. Если n = 180 градусов, то перо выведет полуокружность, при 360 градусах нарисует полную окружность.

Для того чтобы нарисовать точку, используется команда **dot(r, color)**, где r – радиус точки в пикселях, color – цвет, которым будет рисоваться точка.

Задача 1. Нарисуем окружность, по длине которой равномерно распределено заданное количество точек:

```
from turtle import*
def circ(d, r, rBig):  #функция circ с параметрами: кол-во точек, радиус точки, радиус окружности
    for i in range(d):
        circle(rBig, 360 / d) #распределяем кол-во точек по кругу
        dot(r, 'red')

up()
goto(150, 0) #отводим перо на 150 пикселей направо
setheading(90) #поворачиваем перо на 90°
down()          #опускаем перо для начала рисования
circ(15, 10, 150) #передаем значения параметрам
screen.mainloop() #останавливаем выполнение программы
```



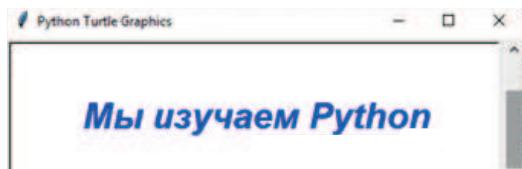
Кроме вывода различных фигур, в окне для графики можно рисовать текст. Для этого используется команда **write()** с множеством параметров:

```
write(text, move, align, font = (fontname, fontsize, fontstyle))
```

- в параметр **text** пишется сам текст в кавычках;
- **align** принимает значения «left», «right», «center» и изменяет положение текста относительно черепашки; значения пишутся в кавычках;
- **font** принимает значения fontname, fontsize, fontstyle:
 - в **fontname** пишется название шрифта в кавычках;
 - **fontsize** отвечает за размер шрифта;
 - **fontstyle** отвечает за стиль текста (**normal** – обычный, **bold** – полужирный, **italic** – курсивный, **bold italic** – полужирный текст).

Задача 2. Напишем программу для вывода форматированного текста:

```
from turtle import*
color ('blue')
write('Мы изучаем Python', 'center', font=('Roboto', 24, 'bold italic'))
>>>
```



Команды **clear()** и **reset()** очищают экран от рисунков и перемещают перо в центр.

Работа с Tkinter для создания графических объектов

Модуль Tkinter для работы с графической библиотекой Tk позволяет создавать программы с более продвинутой графикой и анимацией. Так же как и с модулем turtle, модуль tkinter нужно сначала подключить командой:

```
from tkinter import*.
```

Задача 3. Попробуем для начала создать обычную кнопку. Для этого мы используем виджет **Button**, где в скобках задаем параметры кнопки. С помощью значения **text** устанавливаем надпись на кнопке. Запишем программу:

```
from tkinter import*
tk = Tk() #создадим форму Tk (объект) с именем tk
btn = Button(tk, text='Пуск')
#создадим кнопку с текстом
«Пуск»
btn.pack() #располагаем
кнопку внутри окна
>>>
```



Однако при нажатии этой кнопки ничего не происходит. Для того чтобы нажатие сопровождалось каким-то действием, введем в программу функцию, результат которой выводится через команду **command**:



ЭТО ИНТЕРЕСНО!

Виджеты – это базовые блоки для создания графического интерфейса программы, некоторые из которых стандартны во всех языках программирования. Например, это виджеты кнопок, флагки или полоса прокрутки. Они могут лишь отличаться названиями: например, классические флагки (check box) в Tkinter называются **check button**.



```
from tkinter import*
def btn_act(): #добавляем функцию, которая выводит на консоли
    print('Игра началась!') #слова «Игра началась!»
tk = Tk()
btn = Button (tk, text='Пуск', command=btn_act) #при нажатии
на кнопку выводится сообщение из функции
btn.pack()
```

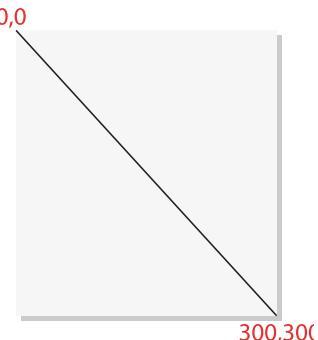
Теперь, после нажатия на кнопку, в консоли выводится сообщение:

```
>>> «Игра началась!»
```

Другой виджет – **Canvas** (англ. холст) позволяет рисовать на заданном холсте. Рисование начинается с указания размеров холста, а именно с указания ее ширины (width) и высоты (height).

Для обозначения стартовой точки рисунка на холсте используются координаты X и Y.

Координаты определяют, насколько пиксель (точка) отступает от левого края холста по горизонтали (X) и от верхнего края холста по вертикали (Y).



Для создания линии используется метод **create_line()**, где в скобках указываются четыре числа. Первые два числа – это координаты начала линии, вторые два – координаты конца линии. Запишем программу:

```
from tkinter import *
tk = Tk()
canvas = Canvas(tk, width=300, height=300)
canvas.pack()
canvas.create_line(0, 0, 300, 300)
```

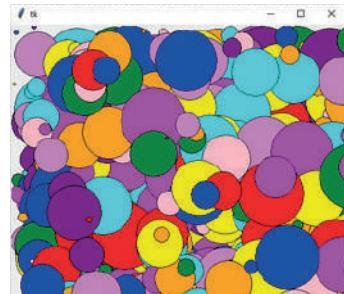
Для создания круга используется метод **create.oval()**:

```
canvas.create_oval(10, 10, 80, 80, outline= 'red', fill=
'green', width=2)
```

В данной записи первые 4 параметра определяют ограничивающие координаты фигуры. Другими словами, это x и y координаты верхней левой и правой нижней точек квадрата, в который помещен круг.

Задача 4. Напишем программу для создания на холсте кругов, диаметры и цвета которых выбираются случайным образом.

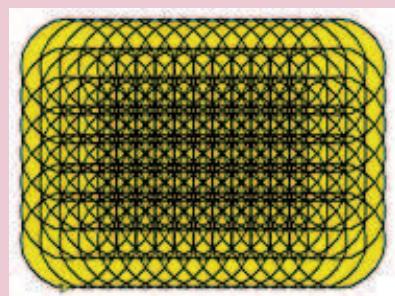
```
from random import* #загружаем функцию randint и choice из
модуля random
from tkinter import*
size = 500 #вводим переменную size
tk = Tk()
diapason = 0 #вводим переменную diapason для ограничения
кол-ва кругов
my_canvas = Canvas (tk, width=size, height=size) #создаем
холст, используя значение переменной size
my_canvas.pack() #расположаем холст внутри окна
while diapason <1000: #цикл повторяется до этого условия
    color = choice(['green', 'red', 'blue', 'orange',
'yellow', 'pink', 'purple', 'violet', 'magenta', 'cyan'])
#создаем список для случайного выбора цветов кругов
    x1 = randint(0,size) #случайный выбор координат x и y
    y1 = randint(0,size)
    d=randint(0,size/5) #произвольный выбор диаметра круга,
но не более size/5
    my_canvas.create_oval(x1,y1,x1+d,y1+d,fill=color) #создаем
круги и заливаем случайным цветом
    tk.update()
    diapason+=1 #шаг цикла, счетчик
```



КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

1) Нарисуйте с помощью модуля *turtle* ковер из кругов, где круги имеют один цвет, а фон ковра – другой.

2) Напишите с помощью модуля *tkinter* программу, которая выводит на холст линии, толщина и цвет которых выбираются случайным образом.



Глава

Ч



Компьютерные сети и интернет

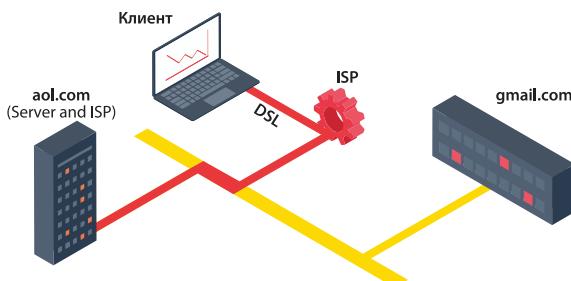
**4.1. Тема:**

Компьютерные сети

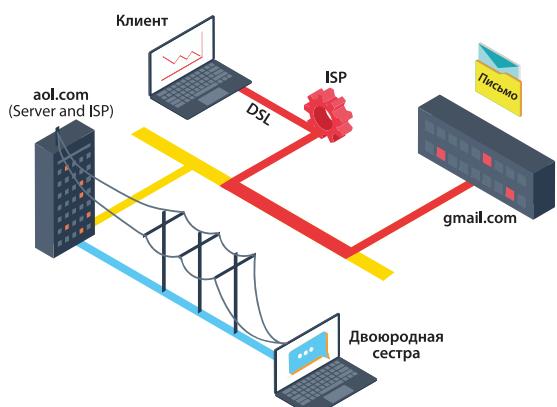
Компьютерная сеть – это система компьютеров, связанная каналами передачи информации. Компьютеры могут быть связаны между собой как с помощью специальных проводов (кабель Ethernet, оптоволоконный кабель), так и без них (Bluetooth, Wi-Fi и др.). Компьютерные сети могут быть локальными (в пределах одного здания) либо глобальными (интернет).

Рассмотрим, как работает глобальная сеть интернет.

Например, мы хотим войти на сайт **aol.com** и прочитать последние новости.



Сначала мы попадаем в сеть провайдера, потом провайдер перенаправляет нас через сеть интернет к сайту, после чего мы можем видеть страницы сайта **aol.com**.



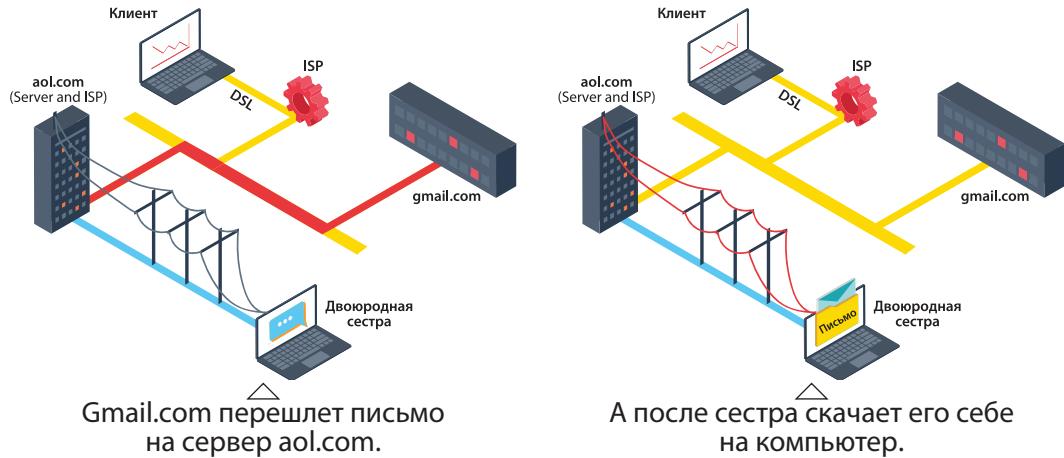
ЭТО ИНТЕРЕСНО!

Для преобразования IP-адресов в имена доменов используются специальные серверы – **DNS** (Сервер имен доменов, *Domain Name Server*).

Эти серверы хранят специальные таблицы сопоставления имен доменов с IP-адресами и по запросу могут преобразовать имена домена в IP-адреса и наоборот – IP-адрес в имя домена. Доменные имена мы вводим в браузере для запроса веб-страниц. Они так же называются **URL** (*Universal Resource Locator*).



Другой пример. Мы хотим отправить электронное письмо двоюродной сестре. У сестры соединение с интернетом через телефонные провода к провайдеру *aol.com*. А у вас почта – *gmail.com*. Почта сестры на сайте *aol.com*. Сначала надо зайти на сайт *gmail.com* в свой почтовый ящик (например, *my@gmail.com*).

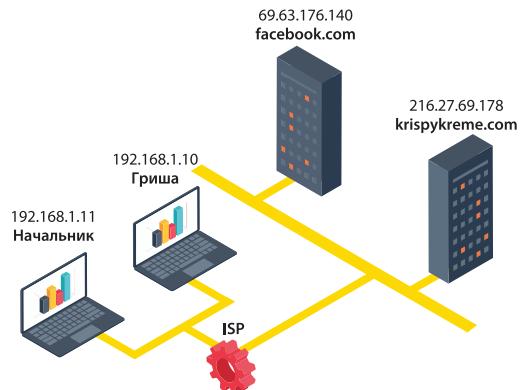


Затем напишать письмо, ввести адрес сестры (*sister@aol.com*) и нажать на кнопку «отправить».

При передаче сообщения в интернет компьютер разбивает сообщение на маленькие части – они называются **пакетами**. На конечном сервере сообщение собирается в нужном порядке. Сообщением может быть не только электронное письмо, но и веб-страница, сообщение Твиттера или любая другая информация.

Допустим, вы пользуетесь интернетом дома (ищете информацию или обновляете профиль в социальной сети) и ваши родители тоже пользуются домашним интернетом. В этом случае вы и ваши родители подключены по Wi-Fi к домашнему маршрутизатору.

Если посмотреть на рисунок, то видно, что ваши пакеты и пакеты ваших родителей идут в интернет через одну точку. То есть и информация, необходимая вашим родителям, и ваша личная переписка с друзьями – все собирается в одном месте.

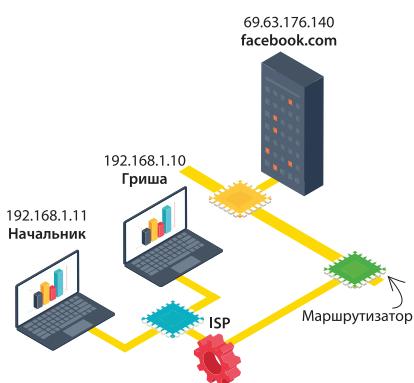




Как сделать так, чтобы ваши личные письма из сети интернет приходили к вам, а не к вашим родителям?

Здесь к нам на помощь приходят IP-адреса и маршрутизаторы. Все, кто подключен к сети интернет напрямую или не напрямую, имеют IP-адреса. Все компьютеры, мобильные телефоны, планшеты, телевизоры и все устройства, которые пересыпают в интернете информацию друг другу, имеют собственный IP-адрес. Устройства, которые соединяют различные части сети интернет, называются маршрутизаторами.

Маршрутизаторы пересыпают пакеты в интернете. Каждый пакет информации направляется к ближайшему маршрутизатору. Если вы заходите на сайт, то обычно ваши пакеты проходят от десяти до двадцати маршрутизаторов, чтобы достичь нужного сайта.



На этой картинке видно, как передвигается сигнал между маршрутизаторами.

Пакеты, передающиеся в сети, состоят из нескольких слоев: первый слой – это IP-адрес вашего компьютера; следующий слой – это IP-адрес вашего маршрутизатора; далее записываются IP-адреса всех маршрутизаторов, через которые проходит пакет. Адреса как бы обертывают друг друга. Это делается для того, чтобы знать, куда возвращать ответ от сервера, на который вы зашли.



ЗАПОМНИ

- **IP-адрес** – это номер, назначающийся каждому устройству, которое присоединяется к сети интернет.
- **DNS (Domain Name Service)** – это сервис, который переводит имена сайтов в IP-адреса.
- **URL (Universal Resource Locator)** – это универсальный указатель сайта, легкий для запоминания адрес (например, www.code.org).
- **Интернет (Internet)** – это группа компьютеров и серверов, соединенных общей сетью друг с другом.
- **Серверы (Servers)** – это компьютеры, которые передают данные и отвечают на запросы других компьютеров.
- **Оптоволоконный кабель (Fiber optic cable)** – это специальный кабель, в котором для передачи информации используется свет.
- **Wi-Fi** – это метод передачи цифрового сигнала, в котором используются радиоволны.
- **DSL** – это метод передачи информации с помощью телефонного или телевизионного кабеля.
- **Пакеты (Packets)** – это маленькие порции информации, которые формируются из больших пакетов информации для более аккуратной передачи сигнала.

Настройка локальной сети и сети Wi-Fi



Локальные сети предназначены для обмена информацией между компьютерами, расположенными на небольшом расстоянии друг от друга, в пределах одного здания.

Попробуем настроить домашнюю (локальную) сеть Wi-Fi. Для этого вам понадобится роутер, который необходимо подключить к вашему провайдеру. Роутер можно настроить либо по кабелю, либо по Wi-Fi.

При подключении по кабелю: кабель с одной стороны подключается в компьютер, с другой – в LAN разъем роутера. Далее в WAN разъем роутера нужно подключить кабель вашего интернет-провайдера.

Для настройки роутера по Wi-Fi достаточно с компьютера подключиться к Wi-Fi сети, которая появится сразу после включения питания на роутере (название сети будет включать марку вашего роутера) и будет не защищена. К этой сети можно также подключиться с планшета или телефона, и настроить маршрутизатор без компьютера. Сам роутер при этом должен быть подключен к сети провайдера через WAN-кабель.

Далее вам нужно зайти в настройки роутера через браузер по адресу 192.168.1.1, или 192.168.0.1. Появится окно запроса имени пользователя и пароля. Введите логин и пароль, указанные внизу роутера на наклейке.

Откроются настройки роутера.

WAN Connection Type: PPTP/Russian PPTP

User Name: username

Password:
Connect Disconnec Disconnected

Serwer IP Address/Name:

Dynamic IP Static IP

IP Address: 0.0.0.0

Subnet Mask: 0.0.0.0

Gateway: 0.0.0.0

DNS: 0.0.0.0 0.0.0.0

Internet IP Adress: 0.0.0.0

Internet DNS: 0.0.0.0 0.0.0.0

MTU Size (in bytes): 1420 (The default is 1420, do not change unless necessary.)

Max Time: 15 minutes (0 means remain active at all times.)

WAN Connection Mode: Connect on Demand Connect Automatically Connect Manually

Save

В выпадающем списке WAN Connection Type выберите тип соединения, которое использует ваш интернет-провайдер. Обычно это PPoE.

Далее введите в полях Username и Password – логин и пароль, выданные вашим провайдером.

Выберите пункт Dynamic IP.

Для сохранения настроек на каждом этапе нажимаем кнопку Save.



Для настройки Wi-Fi нужно открыть на странице настроек вкладку Wireless - Wireless Settings.

Quick Setup
QSS
Network
Wireless
Wireless Settings
Wireless Security
Wireless MAC Filtering
Wireless Advanced
Wireless Statistics
DHCP
Forwarding
Security
Parental Control
Access Control
Advanced Routing
Bandwidth Control
IP & MAC Binding
Dynamic DNS

Wireless Settings

Wireless Network Name: help-wifi.com (Also called SSID)

Region: Kyrgyzstan

Warning: Ensure you select a correct country to conform local law. Incorrect settings may cause interference.

Channel: Auto

Mode: 11 bgn mixed

Channel Width: Auto

Max Tx Power: 300Mbps

Enable Wireless Router Radio
 Enable SSID Broadcast
 Enable WDS Bridging

Save

В поле Wireless Network Name придумать и записать имя для вашей Wi-Fi сети

В меню Region выбрать страну, где вы живете.

Не забудьте сохранять настройки, нажав на кнопку Save.

Чтобы защитить нашу беспроводную сеть паролем, перейдите на вкладку Wireless – Wireless Security. Там нужно выделить тип безопасности WPA/WPA2 – Personal. В поле PSK Password придумайте и запишите пароль, который будет защищать вашу Wi-Fi сеть.

Не забудьте сохранять настройки, нажав на кнопку Save.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Напишите все серверы, через которые проходит ваш пакет до сервера *google.com*, когда вы его открываете в браузере.
- 2) Напишите, какие DNS серверы предоставляет ваш провайдер при соединении роутера с ним.
- 3) Алмаз написал на бумажке IP-адрес, который позже случайно порвал на кусочки. В результате получилось 4 обрывка с фрагментами IP-адреса. Эти фрагменты обозначены буквами А, Б, В и Г. Восстановите IP-адрес. В ответе укажите последовательность букв, обозначающих фрагменты, в порядке, соответствующем IP-адресу.

А.

Б.

В.

Г.

.64

3.13

3.133

20

4.2. Тема:

Виды интернет-протоколов

Протоколы сети интернет и работа в сети

Простое подключение одного компьютера к другому – шаг, необходимый для создания сети, но одного этого шага мало. Чтобы начать передавать информацию, нужно убедиться, что компьютеры «понимают» друг друга, что они способны «общаться». Как же компьютеры «общаются» по сети? Чтобы обеспечить эту возможность, были разработаны специальные средства, получившие название «протоколы».

Понятие протокола применимо не только к компьютерной индустрии. Даже те, кто никогда не имел дела с интернетом, скорее всего сталкивались в повседневной жизни с устройствами, функционирование которых основано на использовании протоколов. Обычная телефонная сеть общего пользования тоже имеет свой протокол, который позволяет аппаратам устанавливать факт снятия трубки на другом конце линии или распознавать сигнал о разъединении. Гудки в телефонной трубке – это язык телефонов.

У компьютеров тоже есть единый язык – **протокол**.

Основные протоколы, используемые в работе интернета:

- **TCP/IP** – передача пакетов;
- **POP3/SMTP** – прием/передача электронных писем;
- **FTP** – передача файлов;
- **HTTP/HTTPS** – передача веб-страниц;
- **IMAP4** – прием/передача электронных писем.

Протокол TCP/IP

Протокол IP представляет собой основу протоколов TCP/IP, который относится к типу протоколов без установления соединения. Протокол IP не гарантирует надежной доставки сообщений, потому что доставка информации не подтверждается.



ОПРЕДЕЛЕНИЯ

Протокол – это правила, в соответствии с которыми происходит передача информации через сеть.



Поток данных протокол IP разбивает на определенные части – **IP-пакеты** или **дейтаграммы** и рассматривает каждый IP-пакет как независимую единицу, не имеющую связи с другими IP-пакетами. Основной задачей протокола IP является передача IP-пакетов между сетями.

Так как один протокол IP не гарантирует надежную доставку сообщений, эту задачу решает протокол TCP. Протокол TCP устанавливает логическое соединение между компьютерами. Перед передачей данных посыпается запрос на начало сеанса передачи, получателем посыпается подтверждение, если через определенное время подтверждения нет, то запрос посыпается повторно.

Части, на которые протокол TCP разбивает поток данных, принято называть **сегментами**. В каждом сегменте есть заголовок, в котором существует поле контрольной суммы.

- Если при пересылке данные повреждены, то по контрольной сумме протокол TCP может это определить. Поврежденный сегмент уничтожается, а источнику ничего не посыпается.
- Если данные не были повреждены, то они пропускаются на сборку сообщения в приложение, а источнику отправляется подтверждение.

Для транспортировки сегментов протокол TCP/IP помещает сегменты в оболочку IP-пакета.

Протокол HTTP

Протокол HTTP (Hypertext Transfer Protocol – Протокол передачи гипертекста) был разработан для эффективной передачи по интернету веб-страниц. Именно благодаря HTTP мы имеем возможность созерцать страницы сети во всем великолепии. Протокол HTTP является основой системы World Wide Web.

Вы отдаете команды HTTP, используя интерфейс браузера, который является HTTP-клиентом. При щелчке мышью на ссылке браузер запрашивает у веб-сервера данные того ресурса, на который указывает ссылка, – например, очередной веб-страницы.

ЗАПОМНИ

Протоколы сети интернет делятся на уровни, для работы одних протоколов требуются другие протоколы. Например, протоколы HTTPS/HTTP, FTP, SMTP, POP3, IMAP используют для передачи данных между компьютерами протокол TCP/IP.

Чтобы текст, составляющий содержимое веб-страниц, отображался на них определенным образом – в соответствии с замыслом создателя страницы, он размечается с помощью особых текстовых меток – тегов языка разметки гипертекста (HyperText Markup Language, HTML).

Необходимо отметить, что существует еще протокол **HTTP-S (HTTP Secure)** – сетевой протокол защищенной передачи гипертекста. В этом случае вся передаваемая информация между веб-сервером и пользователем шифруется, что помогает избежать передачи незашифрованных данных пользователя (имен, паролей, номеров, кредитных карт и т.д.) по сети интернет.

Адреса ресурсов интернета, к которым вы обращаетесь по протоколу HTTP, выглядят примерно следующим образом – <http://www.code.org>.

Протокол FTP

Протокол FTP (File Transfer Protocol – Протокол передачи файлов) позволяет передавать файлы . Передача файлов по протоколу FTP происходит между **FTP-сервером и FTP-клиентом**. FTP-сервер обрабатывает запросы, приходящие от FTP-клиентов – программ, через которые происходит загрузка файлов.

Для начала процесса передачи файла FTP-клиенту требуется передать серверу имя пользователя и пароль. Эти обязательные параметры соединения позволяют серверу идентифицировать пользователя. FTP-сервер можно настроить так, чтобы пользователь мог заходить без идентификации. В этом случае пользователь называется **анонимным** (anonymous), то есть сообщает серверу всегда одно и то же имя пользователя – anonymous. Пароль в этом случае сервером игнорируется.

Дальше мы более подробно рассмотрим протоколы, без которых невозможна работа электронной почты: SMTP, POP3 и IMAP. Эти протоколы используются только для одной цели – для организации обмена электронными сообщениями. Причем, протокол, умеющий отсылать сообщения, не умеет их принимать, и наоборот. Именно поэтому такие протоколы работают парами.

Протокол SMTP

Протокол **SMTP** (Simple Mail Transfer Protocol – Протокол простой почтовой передачи) обеспечивает отправление электронного сообщения. SMTP сер-



вер не способен накапливать сообщения на стороне получателя. Поэтому при получении почты необходим еще один из почтовых протоколов – протокол POP3 или IMAP.

Протокол POP3

Протокол **POP3** (Post Office Protocol 3 – Протокол почтовой службы 3) обеспечивает получение электронного сообщения адресатом. В соответствии с ним почта принимается сервером и накапливается на нем. Программа – почтовый клиент – периодически проверяет почту на сервере и загружает сообщения на локальный компьютер.

Таким образом, отправление почты осуществляется с помощью SMTP, а прием – с помощью POP3. Вот почему в процессе создания учетной записи почты необходимо вводить названия как сервера SMTP, так и сервера POP3.

Протокол IMAP

IMAP (Internet Message Access Protocol – Протокол доступа к электронной почте интернета). Аналогично POP3, служит для работы с входящими письмами, однако обеспечивает дополнительные функции, в частности, возможность поиска по ключевому слову без сохранения почты на локальном компьютере. Почтовая программа, использующая этот протокол, получает доступ к хранилищу корреспонденции на сервере так, как будто эта корреспонденция расположена на компьютере получателя. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без постоянной пересылки с сервера и обратно файлов с полным содержанием писем. Для отправки писем используется протокол SMTP.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Какие протоколы использует браузер, когда вы читаете и отправляете почту в своем почтовом ящике?
- 2) Какие из сайтов, которыми вы пользуетесь, не поддерживают протокол HTTPS?

4.3. Тема:

Каскадные таблицы стилей (CSS)

Обычный HTML позволяет задавать цвет и размер текста с помощью тегов форматирования. Если понадобится изменить параметры однотипных элементов на сайте, придется просматривать все страницы, чтобы найти и поменять теги. Можно использовать другую технологию – CSS (Cascading Style Sheets – каскадные таблицы стилей), которая позволяет хранить цвет, размеры текста и другие параметры в стилях.

Стилем называется набор правил форматирования, который применяется к элементу HTML документа, чтобы быстро изменить его внешний вид. Стили позволяют одним действием применить форматирование к целой группе схожих элементов, например, изменить вид всех заголовков.

Стили предлагают больше возможностей для форматирования, нежели простой HTML, а также могут храниться во внешнем файле. Браузер сохраняет такие документы локально у пользователя (кэширует), поэтому загрузка сайта будет происходить быстрее.

Все доступные свойства форматирования элементов в документе HTML разбиты на 9 категорий:

- **Шрифт** – устанавливает типографские свойства шрифтов.
- **Цвет и фон** – определяет цвет текста и фона, а также картинки в качестве фона.
- **Текст** – определяет выравнивание, форматирование, разрядку текста.
- **Блок** – свойство форматирования блоковых элементов.
- **Визуальное форматирование** – свойства, связанные с блоками отображения элементов, их позиционированием и отображением списков.
- **Фильтры и переходы** – определяют мультимедийные эффекты и преобразования графических изображений.
- **Печать** – определяет спецификацию разрыва страницы.
- **Псевдоклассы** – включаются свойства: @import, cursor, important.
- **Все остальные свойства.**



Синтаксис в CSS

Стиль CSS состоит из селектора (элемента HTML документа), который всегда располагается в левой части блока объявления стилей и заключается в фигурные скобки.



В данном примере применили новый стиль оформления к тегу H1, он теперь будет синего цвета и шрифт у него будет Verdana.

Селектор – это конструкция, позволяющая выбрать элемент, к которому будут применены данные стили.

Наиболее часто используются четыре вида селекторов:

- по элементу;
- по классу;
- по ID;
- контекстный селектор.

Селектор по элементу

Если нужно применить стиль ко всем заголовкам первого уровня, то в качестве селектора используется элемент `<H1>`. Например, чтобы все заголовки первого уровня были темно-серого цвета, выводились шрифтом Arial и имели расстояние между буквами 10 пикселей, пишется такой стиль:

```
h1{color:#666;font-family:Arial;letter-spacing:10px}
```

Селектор по классу

Существует специальный атрибут `class`, который можно использовать в любом элементе для того, чтобы применить конкретные свойства определенному тегу.

Например, чтобы ссылки в меню сделать особенными, вводим новый `class="menu"`.

```
a.menu {font-style: italic;font-weight: bold}
```

HTML код будет таким:

```
<a href="about.html" class="menu">О компании</a> (Особенная ссылка)
```

```
<a href="links.html">Ссылки</a> (Обычная ссылка)
```

Селектор по ID (идентификатору)

Если в селекторе по классу для разделения названия элемента и класса используется точка, то для селектора по ID используется знак **#**. Название элемента можно опустить, и тогда данный стиль можно будет применять к любому элементу с идентификатором **first**.

```
a#first {font-style: italic; font-weight: bold}  
<a href="about.html" id="first">О компании</a>
```

Контекстный селектор

Контекстные селекторы позволяют применять единый стиль для группы элементов, находящихся внутри другого элемента. Например, таблица – это элемент, а все столбцы – это группа элементов.

В данном примере для всех ссылок группы, выделенных тегом **div**,

```
<div id="first">  
  <a href="index.html">Главная страница</a>  
  <a href="about.html">О компании</a>  
</div>
```

будет применяться стиль с идентификатором **first**:

```
a#first {font-style: italic; font-weight: bold}
```

Добавление стилей к документу

Всего существуют четыре способа добавления стилей:

- Вложение Стилей в HTML документ.
- Ссылка из HTML документа на внешний документ Стилей.
- Импортирование документа Стилей в HTML документ.
- Добавление Стилей непосредственно в строки HTML документа.



Вложение Стилей в HTML документ

Вся информация, связанная с CSS, располагается в шапке HTML документа и не контактирует с содержимым тега <BODY>.

Пример:

```
<html>
  <style type="text/css">
    <!--
      h1{color:green;font-family:Impact}
      p{background:yellow;font-family:'Courier New'}
    -->
  </style>
  <head>
    <title>Вложенные стили</title>
  </head>
  <body>
    <h1>Пример 1</h1>
    <p>Пример 2</p>
  </body>
</html>
```

Вложенные стили применяются только для текущего HTML документа. Если вы хотите добавить Стили только на единственную страницу, этот способ подойдет как нельзя лучше.

Импорт стилей

Импортированные стили размещаются в отдельном файле. Их можно объединять со своими собственными стилями, содержащимися внутри документа.

Пример:

```
<html>
  <style type="text/css">
    <!--
      @import url(mytypes.css)
      h1{color:green;font-family:Impact}
    -->
  </style>
  <head>
    <title>Вложенные стили</title>
```

```
</head>
<body>
  <h1> Пример 1</h1>
  <p>Пример 2</p>
</body>
</html>
```

Файл mystyles.css выглядит так:

```
h1{color:orange;font-family:Impact}
p{background:yellow;font-family:'Courier New'}
```

В этом примере браузер сначала импортирует стили из файла mystyles.css (строка @import всегда должна быть первой), а затем добавляет к этому вложенные команды стилей. На странице применяются и те, и другие стили.

Добавление стилей непосредственно в тело HTML документа

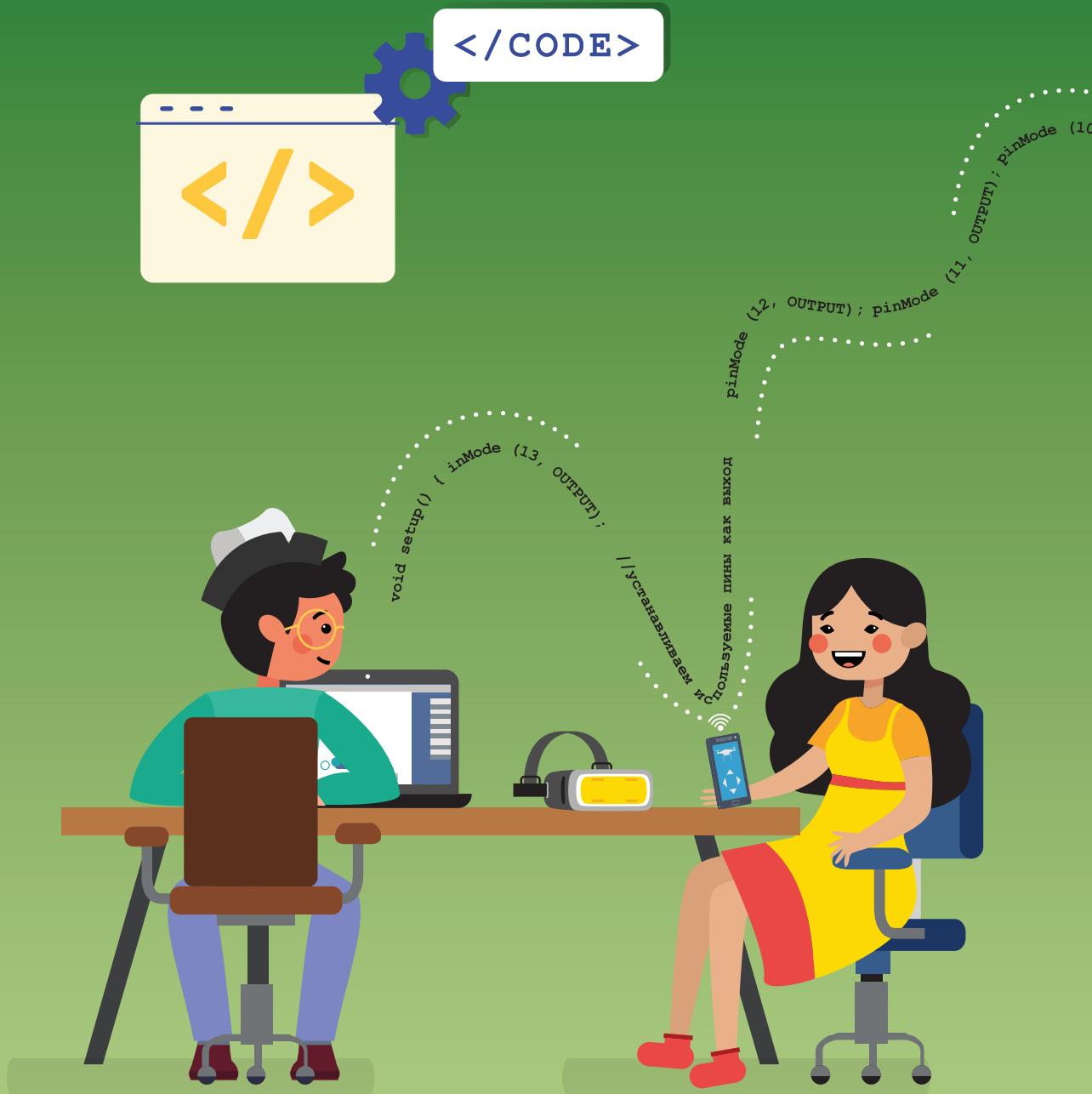
В данном случае не нужно добавлять стили в заголовок HTML документа. Встроенные атрибуты стилей элемента уже содержат всю необходимую информацию для отображения в браузере. Этот стиль будет применяться только для данного элемента.

Пример:

```
<html>
  <head>
    <title>Вложенные стили</title>
  </head>
  <body>
    <h1 style="color:green;font-family:Impact">Пример 1</h1>
    <p style="background:yellow;font-family:'courier'">Пример
2</p>
  </body>
</html>
```

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

Создайте веб-страницу «Я и космос» и импортируйте собственный CSS с определением стилей для заголовков и ссылок.





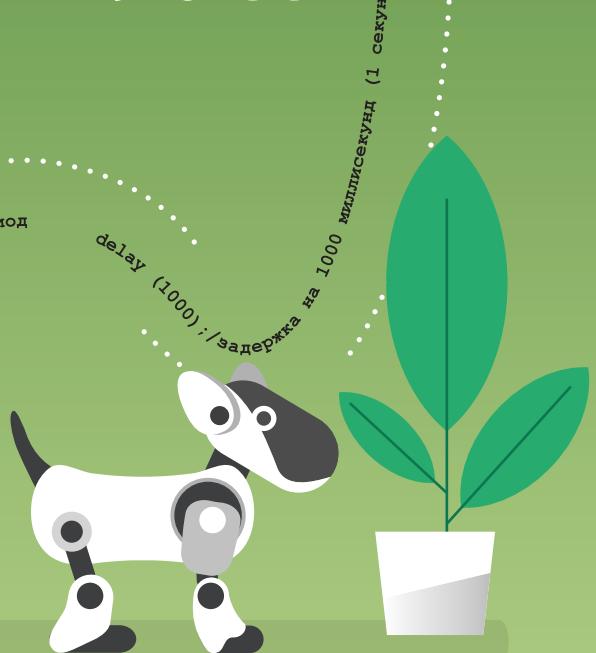
digitalWrite

delay (1000);



класс

```
void loop() {  
    digitalWrite (13, 1);  
    //включаем светодиод  
    delay (1000);  
    digitalWrite (13, 0); //выключаем светодиод  
    delay (1000); //задержка на 1000 миллисекунд (1 секунда) digitalWrite (12, 1);  
    delay (1000);
```



</>

Глава 1



Информатика и информация

Тема 1.1.

Информационная грамотность

Сегодня, когда поток информации увеличивается с каждым днем, умение правильно оценивать и отбирать нужную информацию становится очень важным навыком для любого человека. Этот навык называется информационной грамотностью и отличается от компьютерной грамотности, которая подразумевает умение использовать компьютерные приложения для решения поставленных задач.

Информационная грамотность позволяет:

- Определять потребность в конкретной информации.
- Находить источники информации.
- Подбирать или получать информацию.
- Анализировать и оценивать качество информации.
- Упорядочивать и хранить информацию.
- Этично и эффективно использовать информацию.

При поиске информации важную роль играет умение критически оценивать информацию. Получая информацию в интернете, ответьте себе на 5 вопросов:

- Кто автор статьи, является ли он экспертом в той теме, о которой пишет?
- Соответствует ли содержание статьи тематике всего сайта?
- Есть ли информация о том, кем был создан сайт?
- Видна ли дата публикации статьи?
- Почему я считаю, что этой информации можно верить?

Особенно важно уметь правильно оценивать информацию, которую мы получаем из так называемых «новых медиа» – интерактивных средств массовой информации (социальные сети, новостные каналы и пр.).

ОПРЕДЕЛЕНИЯ

Информационная грамотность – это способность человека осознавать необходимость в информации, умение ее искать, отбирать, оценивать и использовать.

Основными целями таких медиа является привлечение внимания как можно большего числа пользователей и оказание серьезного влияния на мнение аудитории. Для этого используются различные методы: подача серьезных материалов в развлекательной форме либо публикация материалов, содержащих шокирующий контент, который не всегда соответствует действительности. При этом информацией зачастую манипулируют в собственных интересах.

ИНФОРМАЦИЮ МОЖНО

Сфабриковать, выдавая ее за подлинную

Искажить путем неполной, односторонней подачи

Отредактировать, добавив собственные домыслы и комментарии

Интерпретировать в выгодном для манипулятора свете

Утаить, скрыв какие-либо существенные детали

Любая манипуляция информацией приводит к тому, что у читателя складывается ложное мнение.

Возможность, которую дают социальные сети любому человеку стать каналом информирования, приводит к появлению большого количества фейков.

ОПРЕДЕЛЕНИЯ

Фейк (от англ. *fake* – «поддельный, фальшивый») в случае с подачей информации имеет смысл – «ложная информация».

Что сегодня называют «фейками»:

Фотографии, подделанные в фотошопе, видеоролики, специально смонтированные с целью введения в заблуждение, либо иллюстрирующие другое событие. Например, когда иллюстрирующее видео было снято несколько лет назад при проведении военной операции, а его публикуют, указывая, что это видео – свидетельство событий последних дней.

Фальшивые новости, которые не все способны отличить от правды (то, что раньше называлось «газетными утками», сегодня именуют «вбросами»).

Страницы в социальных сетях, созданные от имени других (как правило, известных) людей.



Целью создания фейков в большинстве случаев является мошенничество. Оно может касаться как сбора денежных средств (например, «розыгрыши» смартфонов), так и увеличения продажи определенных товаров путем

публикаций фальшивых отзывов. Также путем публикации фейков мошенники добиваются большого числа просмотров (генерируют трафик). Например, с помощью кричащих заголовков: «Сенсация: английский язык за 1 месяц!», «Она похудела на 20 кг за 2 недели, и ты сможешь!» и пр.

Спам

Другим распространенным «информационным мусором» является спам. Спам – это нежелательная рекламная рассылка, где зачастую отправителя установить невозможно. Такие сообщения распространяются по электронной почте, через социальные сети, форумы, комментарии и СМС-сообщения. По статистике, 90% всех электронных писем сегодня составляет спам. То есть на одно полезное письмо приходится десять писем со спамом. Данные получателей для своих спам-рассылок спамеры (те, кто рассыпает спам) собирают с различных сайтов, где вы когда-либо вводили данные своей электронной почты.

Обычно спам провоцирует вас переходить по ссылкам или открывать подозрительные приложения. И несмотря на то, что сам по себе спам довольно легко удалить, случайный щелчок по ссылке в спаме может привести к серьезным проблемам: заражению вирусом или краже ваших личных данных.

Чтобы не идти на поводу у интернет-мошенников, необходимо уметь мыслить критически, то есть любую информацию подвергать сомнению, и стараться найти первоисточник этой информации.

И самое главное – не принимать все сказанное на веру, особенно в интернете.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Что такое «фейк»? Является ли новость о том, что начало учебного года перенесут на 15 сентября «фейком»? Почему?
- 2) Отличаются ли «фейки» в социальных сетях от телевизионных «фейков»? Объясните, чем они могут отличаться?
- 3) Проведите свое мини-исследование: расскажите свою «фейковую историю» 10 друзьям и замерьте, сколько из них поверили вам.
- 4) Составьте таблицу: какие виды спама бывают, и какой вред может нанести каждый из них.

Тема 1.2.

Шифрование и электронно-цифровая подпись

В цифровую эпоху особую важность приобретают электронные документы.

Электронный документ – это документ, зафиксированный на электронном носителе. Так же как и любые бумажные юридически значимые документы, такие как договоры или справки, электронные документы тоже можно подписывать. Для этого используется электронно-цифровая подпись.

Электронные документы имеют существенные преимущества перед бумажной формой:

- Быстрый срок обработки и доставки электронных документов (можно удаленно подписать любой документ и отправить по email).
- Сокращение расходов: не нужно печатать документ на бумаге, оплачивать доставку по почте и пр.

То есть раньше многие операции, связанные с передачей денег или каких-то прав, например, купля-продажа дома или получение справки от государственного органа проводились только при личном присутствии гражданина в органе, выдающем документ. При этом нам приходилось ставить ручкой свои подписи на целой кипе бумаг. Операторы просили предъявить оригинал паспорта или свидетельства о рождении для того, чтобы идентифицировать личность.

В цифровую эпоху большинство такого рода операций можно осуществлять удаленно через интернет. Главное условие – оператор должен аутентифицировать поступающую от вас информацию. Для этого используются разные технологии, к примеру, для обналичивания денег в банкомате требуется ввести 4-значный код, открывающий доступ к карте, а ваша электронная почта требует ваш личный пароль для того, чтобы дать доступ к письмам.



ОПРЕДЕЛЕНИЯ

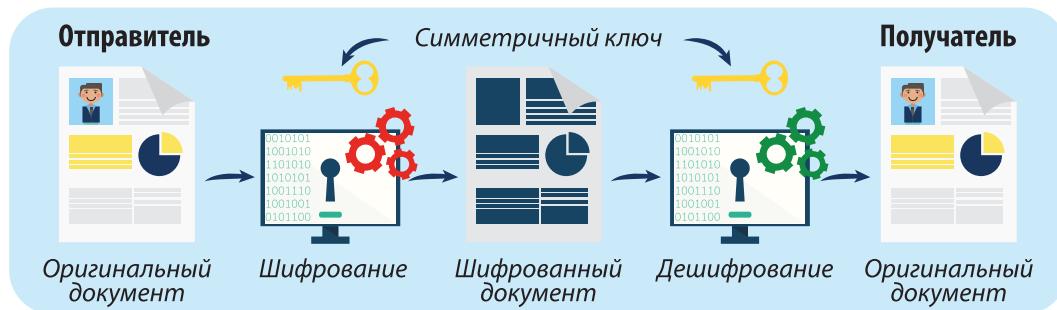
Аутентификация – это процесс проверки подлинности чего-либо. Примером **аутентификации** может быть сравнение пароля, введенного пользователем, с паролем, который сохранен в базе данных сервера.

Однако для подписания электронных документов, например, договоров, заявлений, в которых часто участвует государство, необходимо иметь **электронно-цифровую подпись (ЭЦП)**. ЭЦП – это такая же уникальная подпись, как и подпись ручкой, но используется только для электронных файлов. ЭЦП позволяет установить, кто подписал данный электронный документ, и изменился ли документ после подписания.

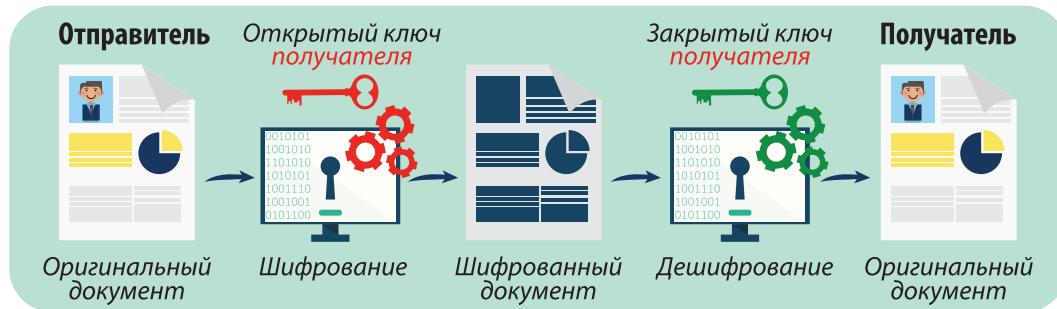
Другими словами, ЭЦП – это механизм шифрования документа.

Исторически использовались два вида шифрования:

- Симметричный – используется один ключ, то есть одним и тем же ключом документ и зашифровывается, и расшифровывается:



- Асимметричный – используется пара ключей: одним ключом документ зашифровывается, другим расшифровывается.



При асимметричном шифровании, если кто-то захочет обменяться с вами зашифрованным сообщением, он берет ваш открытый ключ (который вы даете всем желающим), зашифровывает сообщение и отправляет это сообщение вам.

А вы при получении этого сообщения, используя ваш закрытый ключ (известный только вам), расшифровываете это сообщение.

Таким образом, проверка подписи представляет собой повторное ее вычисление и сравнение с ключом, имеющимся на стороне проверяющего. При проверке выдается сообщение, что подпись либо верна, либо нет. Если подпись неверна, то либо используется неверный ключ, либо сам документ был изменен.

В заключение отметим, что благодаря своей защищенности, асимметричное шифрование получило более широкое распространение. Его используют сайты с поддержкой протокола **HTTPS**, мессенджеры (вспомните, Whatsapp часто пишет, что «ваши сообщения защищены сквозным шифрованием»), wifi-роутеры, банковские системы и многое другое. На основе асимметричной криптографии базируется **электронная подпись**. Также на асимметричной криптографии построен алгоритм **блокчейна**, на котором, в свою очередь, построены все криптовалюты, включая **биткоин**.



ЭТО ИНТЕРЕСНО!

Блокчейн (англ. *blockchain*) – выстроенная по определенным правилам непрерывная последовательная цепочка блоков, содержащих информацию. Чаще всего копии цепочек блоков хранятся на множестве разных компьютеров независимо друг от друга.

Другими словами, блокчейн – это распределенная база данных, где хранилища данных не связаны общим процессором. Все пользователи цепочки могут видеть, что другие пользователи делают с информацией в базе, но изменять они могут только те блоки информации, которые им принадлежат. Это делает все процессы прозрачными, так что подделать какой-то документ в этой базе – невозможно.

Благодаря технологии блокчейн возникла новая криптовалюта – **биткоин**, не имеющая привязки ни к одному мировому банку или экономике какой-либо страны. У нее есть своя стоимость (курс), ее можно покупать и продавать в интернете. Главная особенность биткоина – анонимность.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Назовите преимущества симметричного и асимметричного шифрования.
- 2) Приведите примеры из вашей жизни, где используются симметричная и несимметричная аутентификация.

Тема 1.3.

Кодирование графической информации

По форме хранения графическую информацию разделяют на **аналоговую** (изображения на бумаге) и **цифровую** (компьютерную графику).

Компьютерная графика – это раздел информатики, который изучает методы получения, создания и обработки графической информации с помощью вычислительной техники.

Как и все виды информации, изображения в компьютере закодированы в виде двоичных последовательностей. Кроме этого, каждая точка в цифровом изображении имеет свой код цвета. Чем больше точек и количество используемых цветов, тем качественнее изображение.

По способу создания графические изображения можно разделить на три группы:

Векторные (это изображения, закодированные в виде набора простейших геометрических фигур, параметры которых хранятся в виде чисел, например: размеры, координаты вершин, углы наклона, цвет контура и заливки). Достоинства векторной графики: возможность масштабирования без потери качества изображения, сравнительно небольшой информационный объем.

Фрактальные (в памяти компьютера хранится формула, по которой строится изображение). Достоинства фрактальной графики: простота создания, возможность бесконечного масштабирования – увеличение сложности рисунка практически не влияет на объем файла.

Растровые (формируются с помощью раstra – совокупности пикселей, каждый из которых имеет свои координаты и код цвета). Достоинства растровой графики: точность цветопередачи, реалистичность изображения.



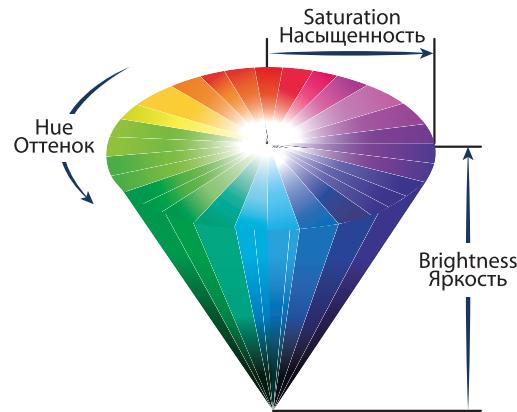
ОПРЕДЕЛЕНИЯ

Дискретизация – это преобразование графической информации из аналоговой формы в дискретную, то есть разбиение непрерывного графического изображения на отдельные элементы.

Изображение из аналоговой формы в цифровую (дискретную) преобразуется путем дискретизации, например, путем сканирования.

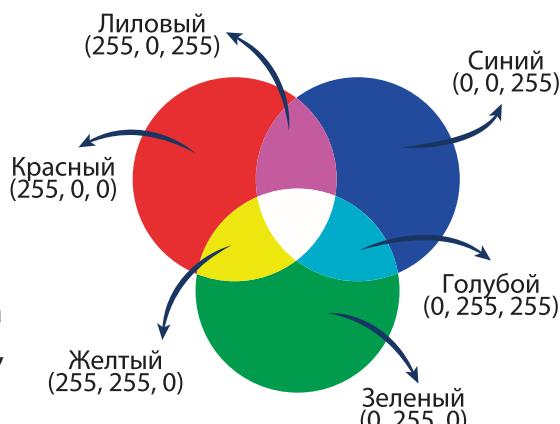
Рассмотрим три основных системы кодирования и передачи цвета для растровых изображений: HSB, RGB и CMYK.

Модель HSB состоит из трех компонентов: оттенок цвета (Hue), насыщенность цвета (Saturation) и яркость цвета (Brightness). Вектор, выходящий из центра окружности, задает значение цвета. Цветовой оттенок определяется направлением вектора и задается в угловых градусах. Насыщенность цвета определяется длиной вектора, а яркость цвета задается на отдельной оси, нулевая точка которой имеет черный цвет. Точка в центре соответствует белому (нейтральному) цвету, а точки по периметру – чистым цветам.



Модель RGB: любой цвет представляется в виде комбинации трех цветов: красного (Red, R), зеленого (Green, G), синего (Blue, B). Другие цвета и их оттенки получаются за счет наличия или отсутствия этих составляющих.

Например, при 256 градациях цвета каждая точка кодируется 3 байтами, потому что на кодирование одного цвета выделяется 1 байт – 8 бит ($2^8=256$). Так как модель использует три цвета, то для кодирования всех возможных сочетаний цветов требуется 3 байта.



Минимальное значение RGB (0,0,0) соответствует черному цвету (минимальная освещенность экрана), белому – максимальное значение RGB (255, 255, 255), а серому – равное значение каждого цвета (например: RGB (35, 35, 35)).

Модель CMYK используется при типографской печати. Дополнительный цвет получают за счет суммирования пары остальных основных цветов или вычитанием из белого.

Дополнительными цветами для красного является голубой (Cyan, C) = зеленый + синий = белый – красный, для зеленого – пурпурный (Magenta, M) = красный + синий = белый – зеленый, для синего – желтый (Yellow, Y) = красный + зеленый = белый – синий. Смешивание цветов модели CMYK не дает чистый черный цвет, поэтому используют дополнительный черный компонент, обозначаемый буквой K (Black).

Различают несколько режимов представления цветной графики:

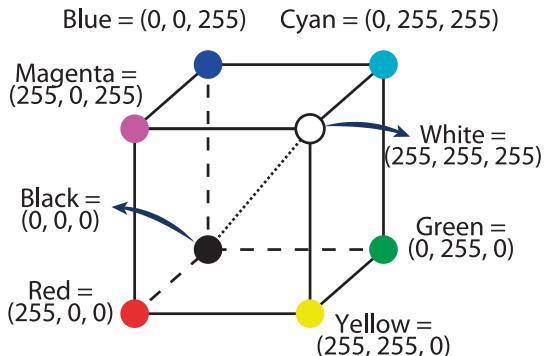
а) полноцветный (True Color) – для кодирования яркости каждой из составляющих используют по 256 значений (восемь двоичных разрядов), то есть на кодирование цвета одного пикселя (в системе RGB) надо затратить $8 \times 3 = 24$ разряда. Это позволяет однозначно определять 16,5 млн цветов. При кодировании с помощью системы CMYK для представления цветной графики надо иметь $8 \times 4 = 32$ двоичных разряда.

б) High Color – это кодирование при помощи 16-разрядных двоичных чисел, то есть уменьшается количество двоичных разрядов при кодировании каждой точки. Но при этом значительно уменьшается диапазон кодируемых цветов.

Соответствие между количеством отображаемых цветов (K) и количеством бит для их кодировки (a) находится по формуле:

$$K = 2^a.$$

Цвета кодируются в двоичном коде. Для кодирования 16-цветного изображения потребуется 4 бита ($16 = 2^4$) для каждого пикселя. А 16 бит (2 байта) используется для кодирования $2^{16} = 65536$ различных цветов. Использование 3 байтов (24-х битов) для кодирования цвета одной точки позволяет отразить 16777216 (или около 17 миллионов) различных оттенков цвета.



Аддитивная система - RGB
Субтрактивная система - CMY

$$\begin{aligned} C &= G+B = W-R \\ M &= R+B = W-G \\ Y &= R+G = W-B \end{aligned}$$

Графическая информация на экране монитора

Качество изображения определяется разрешающей способностью монитора, т.е. количеством точек, из которых оно складывается. Чем больше разрешающая способность, тем выше качество изображения.

Экран монитора может работать в двух основных режимах: текстовом и графическом.

Графические режимы характеризуются такими показателями, как:

- разрешающая способность** – количество точек, с помощью которых на экране воспроизводится изображение, например, 1920x1080.
- глубина цвета** – количество бит, используемых для кодирования цвета точки, например, 8, 16, 24, 32 бита. Количество цветов, отображаемых на экране монитора, может быть вычислено по формуле:

$$K=2^l,$$

где K – количество цветов, l – глубина цвета или битовая глубина.

Таблица. Глубина цвета (l) и количество отображаемых цветов (K)

Глубина цвета (l)	Количество отображаемых цветов (K)
1	2 (черный и белый)
2	$2^2 = 4$
4	$2^4 = 16$
8	$2^8 = 256$
16 (HighColor)	$2^{16} = 65\,536$
24 (True Color)	$2^{24} = 16\,777\,216$
32 (True Color)	$2^{32} = 4\,294\,967\,296$



ОПРЕДЕЛЕНИЯ

Разрешение – это количество пикселей, приходящихся на дюйм размера изображения.
Глубина цвета – это количество бит, используемых для кодирования цвета пикселя.

Качество кодирования изображения зависит от:

- 1 **частоты дискретизации**, т.е. размера фрагментов, на которые делится изображение.
- 2 **глубины кодирования**, т.е. количества цветов для одной точки.

Для того чтобы на экране монитора воспроизвести изображение, необходима видеопамять компьютера, в которой хранятся коды цветов всех точек изображения.

Объем видеопамяти рассчитывается по формуле:

$$V=I*X*Y,$$

где I – глубина цвета отдельной точки,

X, Y – размеры экрана по горизонтали и по вертикали (произведение X на Y – разрешающая способность экрана).

Примеры:

Задача 1. Черно-белое (без градаций серого) растровое графическое изображение имеет размер $10*10$ точек. Какой объем памяти займет это изображение?

Решение:

Количество точек – 100

Так как всего 2 цвета: черный и белый, то глубина цвета равна 1 ($2^1 = 2$)

Объем видеопамяти равен $100*1=100$ бит

Ответ: 100 бит.

Задача 2. Для хранения растрового изображения размером 128×128 пикселей отвели 4 Кб памяти. Каково максимально возможное число цветов в палитре изображения?

Решение:

Определим количество точек изображения. $128*128=16384$ точек или пикселей. Объем памяти на изображение 4 Кб выражим в битах, так как $V=I*X*Y$ вычисляется в битах. $4 \text{ Кб} = 4*1024 = 4096$ байт = $4096*8$ бит = 32768 бит

Найдем глубину цвета $I = V/(X*Y) = 32768:16384 = 2$

$N=2^I$, где N – число цветов в палитре. $N=2^2=4$

Ответ: 4



ОПРЕДЕЛЕНИЯ

Видеопамять – это часть оперативной памяти, в которой формируется графическое изображение.

Задача 3. Определите объем видеопамяти компьютера в килобайтах, который необходим для реализации графического режима монитора High Color с разрешающей способностью 1024 x 700 точек и палитрой цветов из 65536 цветов.

Решение:

1. По формуле $K=2^l$, K – количество цветов, l – глубина цвета.

Так как $2^l = 65\,536$, то по таблице определяем, что глубина цвета составляет:
 $l = 16$ бит ($2^{16} = 65\,536$)

2. Количество точек изображения равно: $1024 * 700 = 716\,800$

3. Требуемый объем видеопамяти равен: 16 бит * $716\,800 = 11\,468\,800$ бит

Переведем в Килобайты:

$11\,468\,800$ бит : $8 = 1\,433\,600$ байт

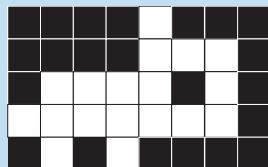
$1\,433\,600$ байт: $1024 = 1400$ Кб

Ответ: 1400 Кб

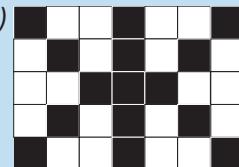
ВОПРОСЫ И ЗАДАНИЯ:

1) Постройте двоичные коды для черно-белых рисунков и запишите их в шестнадцатеричной системе счисления:

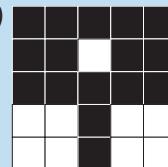
а)



б)



в)



2) Как связаны глубина цвета и объем файла?

3) Какова глубина цвета, если в рисунке используется 65536 цветов? 256 цветов? 16 цветов?

4) Сколько места занимает палитра в файле, где используются 64 цвета? 128 цветов?

5) Поместится ли рисунок размером 800×1000 пикселей, содержащий 1 млн цветов, на флешку объемом 1 Гб?

6) Разрешение экрана монитора – 1024 x 768 точек, глубина цвета – 16 бит. Каков необходимый объем видеопамяти для данного графического режима?

Глава

2



Компьютер и ПО

Тема 2.1.

Компьютерная графика

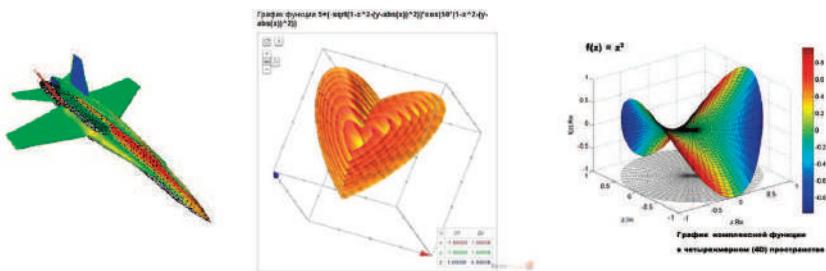
Как мы помним, компьютерная графика – это область компьютерных технологий, которая позволяет создавать, оцифровывать, редактировать визуальную информацию, полученную из реального мира, с целью дальнейшего ее хранения и обработки.

Компьютерную графику применяют представители разных профессий: инженеры и конструкторы, архитекторы и астрономы, дизайнеры и модельеры, художники и ученые. Для каждого из них создается специальное программное обеспечение, которое называют графическими программами или графическими пакетами.

1 Основные области применения

Деловая и научная графика служат для наглядного представления числовых данных в форме графиков, сводок, иллюстраций, а также графической обработки объектов научного исследования.

Примеры:



Виды программ для создания и обработки инженерной графики

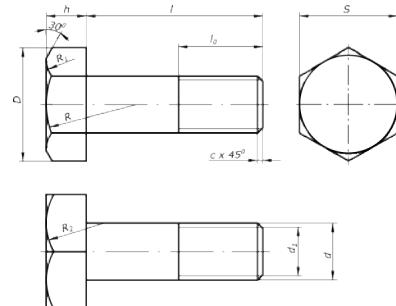
Gnuplot – свободная программа для создания двух- и трехмерных графиков.

Zhu3D – интерактивный, основанный на OpenGL, пакет визуализации математических функций и результатов расчетов. Включает эффекты анимации, изменение текстур, вращение изображений и т.д.

SciDAVis – свободное ПО для анализа и визуализации научных данных, может строить различные типы 2D и 3D-графиков, которые могут сохраняться в нескольких растровых графических форматах файлов, а также в форматах PDF, EPS или SVG. Содержит скриптовый интерфейс для языка Python.

MapViewer позволяет составлять и корректировать карты – изменения масштаб, координаты и т.д., а также обрабатывать и визуализировать демографические данные.

2 Инженерная графика используется в работе инженеров-конструкторов, архитекторов, изобретателей новой техники. Этот вид компьютерной графики является обязательным элементом САПР (систем автоматизации проектирования). Средствами конструкторской графики можно получать как плоские изображения (проекции, сечения), так и пространственные трехмерные изображения.



3 Иллюстративная графика – пакеты иллюстративной графики относятся к ППО общего назначения. Пример – графические редакторы (GIMP, AdobePhotoshop, Adobe Illustrator, CorelDraw и др.)

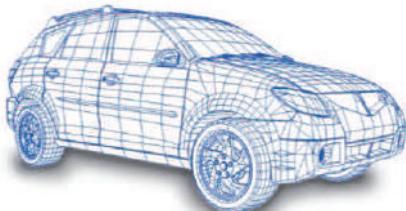
4 Художественная и рекламная графика – с ее помощью создаются рекламные ролики, мультфильмы, компьютерные игры, видеоуроки, видеопрезентации. Графические пакеты для этих целей требуют больших ресурсов компьютера: скорости и памяти. Отличительной особенностью является возможность создания реалистичных изображений и «движущихся картинок». Получение рисунков трехмерных объектов, их повороты, приближения, удаления, деформации связаны с большим объемом вычислений. Передача освещенности объекта, в зависимости от положения источника света, от расположения теней, от фактуры поверхности, требует расчетов, учитывающих законы оптики.

5 Компьютерная анимация – это получение движущихся изображений на экране дисплея. Художник создает на экране рисунки начального и конечного положения движущихся объектов, все промежуточные состояния рассчитывает и изображает компьютер, выполняя расчеты, опирающиеся на математическое описание данного вида движения. Полученные рисунки, выводимые последовательно на экран с определенной частотой, создают иллюзию движения.

По способам задания изображений:

- Двумерная графика (**2D**) – изображения или видео на плоскости.
- Трехмерная графика (**3D**) – изображения или видео, смоделированные объемные объекты в трехмерном пространстве.

3D-моделирование – это разработка объемного образа информационной модели объекта реального мира (автомобиль, здание, молния, астероид) или абстрактных объектов (проекция фрактала – геометрической фигуры, в которой один и тот же мотив повторяется в последовательно уменьшающемся или увеличивающемся масштабе).



Порядок построения трехмерного изображения:

- 1 Моделирование – создание трехмерной математической модели объектов.
- 2 Текстурирование – разработка текстур и свойств материалов (например, прозрачность), воспроизведение растровых, векторных или процедурных текстур.
- 3 Освещение – моделирование виртуальных источников света (направление света, степень освещенности).
- 4 Анимация – создание движения объектов.
- 5 Динамическая симуляция – автоматический расчет взаимодействия различных элементов друг с другом и с моделируемыми процессами (например, ветер, гравитация).
- 6 Визуализация (рендеринг) – отображение трехмерной модели в двухмерном пространстве (на экране монитора).

Используемые программы: Autodesk 3ds Max, Autodesk Maya, Autodesk Softimage, Blender, Cinema, 4D Houdini, Modo, LightWave 3D.

Из указанных программ Blender является свободным ПО для создания трехмерной компьютерной графики.

Blender может использоваться для создания сложных трехмерных моделей и интерактивных игр и включает в себя:

- средства моделирования;
- анимации;
- рендеринга;
- постобработки;
- монтажа видео со звуком;
- компоновки с помощью «узлов».

Программирование в среде Blender осуществляется с помощью языка Python.

Своеобразным расширением 3D-графики является «дополненная реальность» (augmented reality, AR). От «виртуальной реальности», где все окружение построено на компьютерной графике, «дополненная реальность» отличается тем, что в ней компьютерная графика используется лишь частично. К примеру, по такой технологии созданы «Маски» в приложении Инстаграм: вы наводите камеру на свое лицо, и программа сама «подклейивает» на изображение лица мультишные ушки и носик как у собачки, очки либо украшения.



То есть используется технология распознавания изображений (маркеров) в реальной физической среде, к которым программа дополненной реальности достраивает виртуальный 3D-объект. Вы можете изменять маркер: поворачивать в разные стороны, по-разному освещать, закрывать некоторые его части, при этом положения 3D-объектов на экране будут соответственно изменяться.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) К какой компьютерной графике относится визуализация отображения человека в зеркале?
- 2) Чем отличается 3-мерная графика от 3D-моделирования?
- 3) Приведите примеры, где еще можно было бы использовать «дополненную реальность»?

Тема 2.2.

Введение в робототехнику

Мы живем в то время, когда хочется переложить многие домашние и рабочие хлопоты на плечи интеллектуальных помощников. И на помощь приходят роботы.

Робот – это механическое устройство, которое программируемо и в дальнейшем способно выполнять задачи и взаимодействовать с внешней средой без помощи человека.

Робототехника – сфера деятельности, связанная с проектированием, производством и эксплуатацией программируемых механических устройств – роботов.



ЭТО ИНТЕРЕСНО!

Впервые слово «робот» было использовано чешским драматургом Карлом Чапеком в 1921 в произведении «Универсальные роботы Россума», где описывался класс рабов, искусственно созданных человекоподобных слуг, сражающихся за свою свободу. Чешское слово «*robot*» означает «принудительное рабство». А слово «робототехника» было впервые применено известным автором научной фантастики Айзеком Азимовым в 1941 году.



На сегодняшний день роботы стали многофункциональными и имеют массу применений. Области применения роботов делятся на:

- **промышленные роботы** – их использование позволило ускорить и повысить точность многих производственных процессов (упаковка, сборка, окраска и укладка на поддоны). Роботы таких видов принимают решения на основе сложного ответа от датчиков и оснащены системами технического зрения. Они могут использоваться для выполнения сложных, опасных задач, а также задач, которые человек выполнить не в состоянии (обезвреживать бомбы, обслуживать ядерные реакторы, исследовать глубины океана и достигать самых дальних уголков космоса);

- **исследовательские роботы** – используются для выполнения задач в опасной и сложной среде (изучение космического пространства и планет солнечной системы и т.д.);
- **образовательные роботы** – используются для проектирования и решения определенных учебных задач.

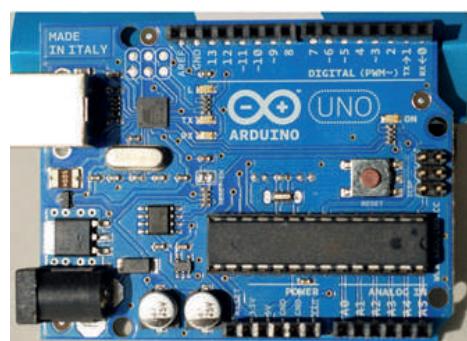


Основные компоненты современных роботов:

- **Механическая составляющая (тело/рама)** – это непосредственно конструкция робота, она может иметь любую форму и размер. Роботы могут быть человекоподобными, но в то же время не имеют ничего общего с человеческим обликом, т.к. в проекте робота меньше внимания уделяется внешности, а больше функциональности.
- **Программная составляющая (система управления)** – управляющая всеми элементами робота. К примеру, датчики робота при взаимодействии с внешней средой отправляют сигналы в центральный процессор, который обрабатывает данные с помощью программного обеспечения и принимает решения на базе логики. То же самое происходит при вводе команд пользователями.

Arduino

Arduino – это одноплатный микрокомпьютер, объединяющий в себе механическую и программную составляющую. Это одна из популярных аппаратных платформ среди школьников, она используется для обучения созданию разного рода роботов. Arduino может использоваться как для создания автономных интерактивных объектов, так и подключаться к программному обеспечению, выполняемому на компьютере.



В робототехнике Arduino часто используется как **мозг робота**.

Плата имеет множество аналоговых и цифровых портов, к которым можно подключить практически любое простое устройство: кнопка, датчик, мотор, экран.

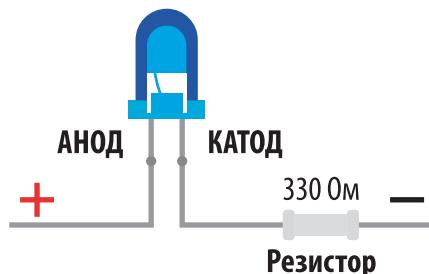
Внутри Arduino уже зашит загрузчик (BootLoader) – программа, которая запускается каждый раз, как только вы включаете аппарат. Для того чтобы запрограммировать Arduino, достаточно подключить плату к персональному компьютеру через USB порт.

Для написания программ используется свободно распространяемый редактор Arduino IDE, который можно скачать здесь:

<https://www.arduino.cc/en/Main/Software>.

Задача 1. Подключим к Arduino светодиод и заставим его мигать.

Это одна из базовых схем в робототехнике. Прежде чем подключим светодиод к плате, давайте его рассмотрим. Светодиод имеет два вывода (ножки), один из которых чуть длиннее другого. Длинный вывод – это анод, который подключают к плюсу источника питания (н-р, батарейки).

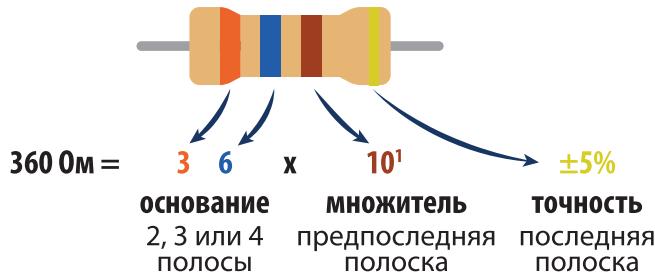


Катод – короткий вывод, который соединяют с минусом или общим проводом.

Резистор – это искусственное «препятствие» для тока. Он ограничивает силу тока, переводя часть электроэнергии в тепло. Выходы резистора не являются ни плюсом, ни минусом, поэтому он является неполярным.

Основной характеристикой резистора является сопротивление. Единица измерения сопротивления – **Ом**. Чем больше сопротивление, тем большая часть тока рассеивается в тепло.

Уровень сопротивления или **номинал резистора** определяют по цветным полоскам вокруг резистора:



Включайте светодиод только через резистор!

Для сборки модели нам потребуется:

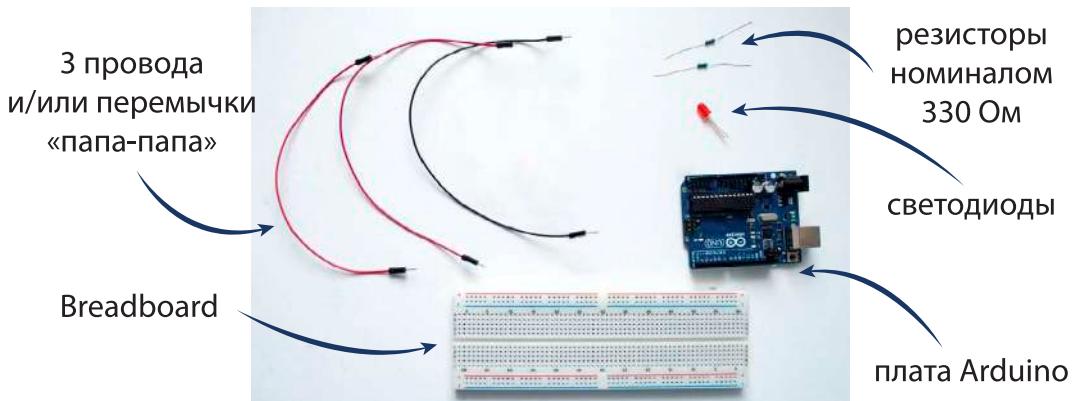
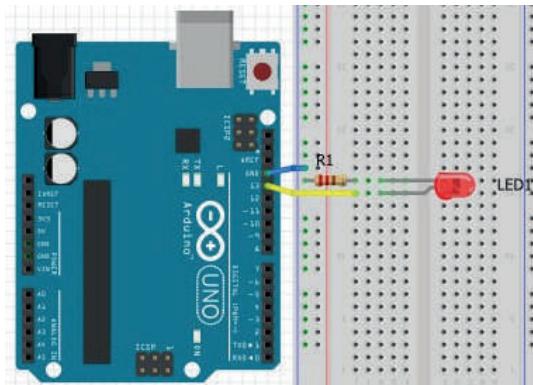


Схема подключения светодиода:



Для работы этой схемы подойдет следующая программа (программу вы можете скопировать в Arduino IDE):

```
void setup() { //обязательная функция setup при старте программы
pinMode (13, OUTPUT); //объявление используемого пина, на выход OUTPUT
}
void loop() { //функция loop запускает программы циклически
digitalWrite (13, HIGH); //команда на включение светодиода
delay (1000); //задержка на 1000 миллисекунд (1 сек.)
digitalWrite (13, LOW); //команда на выключение светодиода
delay (1000); //задержка на 1000 миллисекунд (1 сек.)
}
```

В результате мы увидим, как светодиод будет включаться и выключаться на 1 секунду, т.е. мигать.

Объявление функций в среде программирования Arduino начинается со слов **void**.

Функция **void setup()** является обязательной функцией, запускаемой в начале программы. Блок команд, исполняемых этой функцией, записывается внутри фигурных скобок. Эти команды она выполнит только один раз – в момент старта программы.

В нашем примере внутри **void setup** указана инструкция **pinMode**, которая устанавливает 13-й pin (вход/выход) платы в режим «выход» (OUTPUT).

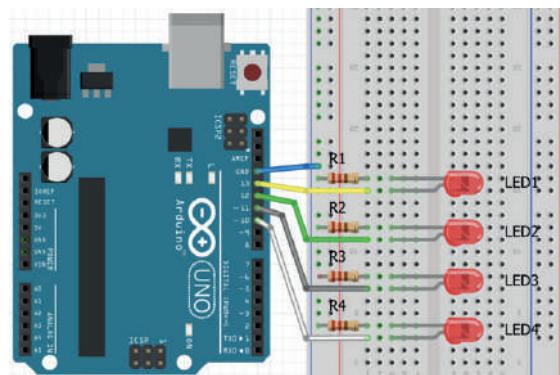
Функция **void loop** вызывается после **void setup** и выполняет команды все время (циклически), пока включена плата Arduino (подается электричество). В нашем примере функция **delay** (задержка на 1 секунду) позволяет нам заметить мигание светодиода. Без нее сигнал включения/выключения будет незаметен.

Задача 2. Давайте теперь добавим в нашу схему еще 3 светодиода. Эта схема подключения называется «схема подключения с общим катодом».

```
void setup() {
  pinMode (13, OUTPUT);
  //устанавливаем используемые пины как выход
  pinMode (12, OUTPUT);
  pinMode (11, OUTPUT);
  pinMode (10, OUTPUT);

}

void loop() {
  digitalWrite (13, 1);          //включаем светодиод
  delay (1000);
  digitalWrite (13, 0);          //выключаем светодиод
  delay (1000);    //задержка на 1000 миллисекунд (1 секунда)
  digitalWrite (12, 1);
  delay (1000);
  digitalWrite (12, 0);
  delay (1000);
  digitalWrite (11, 1);
  delay (1000);
  digitalWrite (10, 1);
  delay (1000);
}
```

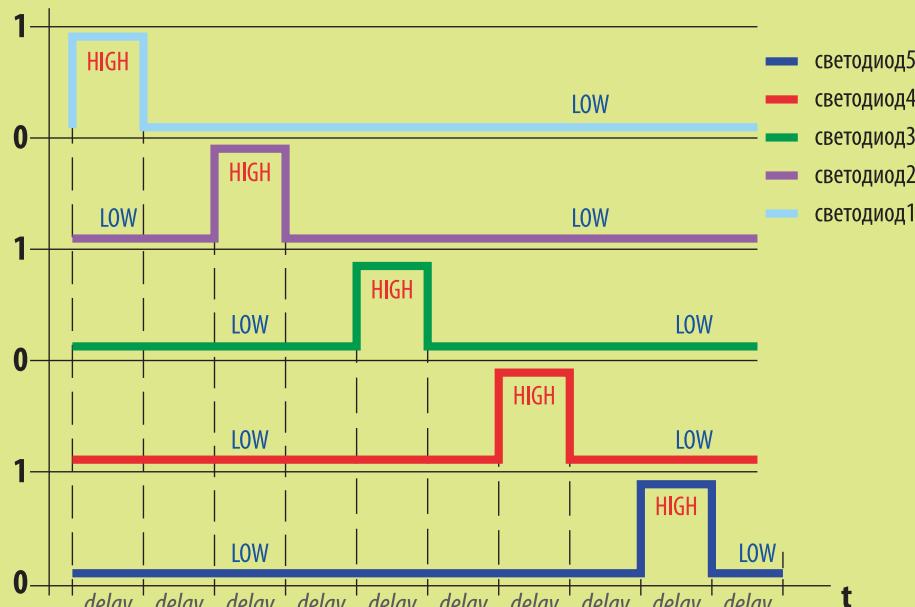


```
digitalWrite (11, 0);
delay (1000);
digitalWrite (10, 1);
delay (1000);
digitalWrite (10, 0);
delay (1000);
}
```

Как вы заметили, мы заменили команды HIGH и LOW на 1 и 0, что никак не повлияло на работу вашей программы.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Как обычно называются конечности робота?
 - 2) Найдите в интернете информацию о 3-х законах робототехники. Расскажите, действовал ли главный герой Арнольда Шварценеггера в фильме «Терминатор» в рамках этих законов?
 - 3) Как сделать так, чтобы все светодиоды включались и выключались одновременно?
 - 4) Соберите схему с подключением пяти светодиодов и напишите программу для бегущих огней согласно следующему алгоритму.



Глава

3



Программирование

Тема 3.1.**Рекурсия**

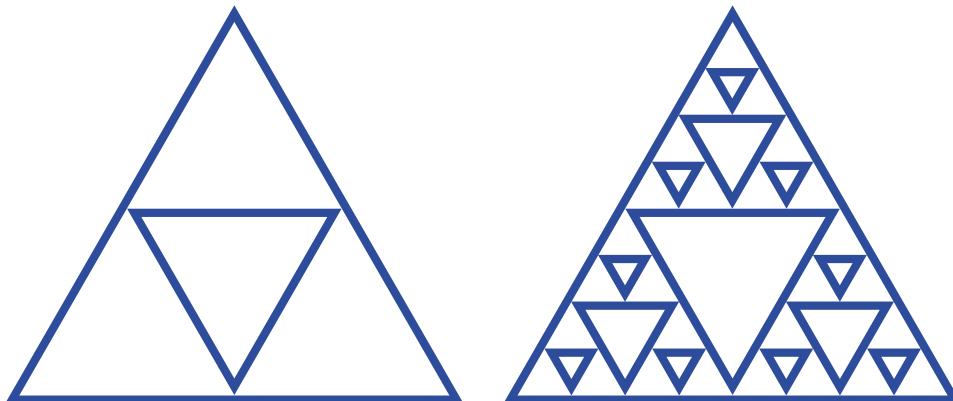
Из курса 8 класса вы помните, что часто одна функция может вызывать другую функцию. Но функция также может вызывать и саму себя. Ситуация, когда программа вызывает сама себя непосредственно (простая рекурсия) или косвенно (через другие функции), например, функция А вызывает функцию В, а функция В – функцию А называется **рекурсией**.



Заметим, что при косвенном обращении все функции в цепочке – рекурсивные.

Популярные примеры рекурсивных объектов – **фракталы**. Так в математике называют геометрические фигуры, обладающие **самоподобием**. Это значит, что они составлены из фигур меньшего размера, каждая из которых подобна целой фигуре.

Рис.1. Треугольник Серпинского (один из первых фракталов, предложенных в 1915 году польским математиком В. Серпинским).



Равносторонний треугольник делится на 4 равных треугольника меньшего размера (левый рисунок), затем каждый из полученных треугольников, кроме центрального, снова делится на 4 еще более мелких треугольника и т.д.

Задача 1. Рассмотрим, как работает рекурсия на примере вычисления факториала для $n!$. Например, чтобы вычислить факториал числа 3, нужно $3*2*1$, то есть число $n*(n-1)*((n-1)-1)$.

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n  
- 1)  
print(factorial(3))  
  
=>  
6 #результат
```



ВСПОМНИ

Факториалом числа n называется произведение всех натуральных чисел от 1 до n :

$$n!=1*2*3*4*...*n$$

Например, факториал числа 5 равен 120 ($5!=1*2*3*4*5$).

Рекурсивные функции решают много задач в программировании, но к сожалению, при их написании часто возникают ошибки. Одна из распространенных ошибок – это бесконечная рекурсия, когда цепочка вызовов функций никогда не завершается и продолжается, пока не закончится свободная память в компьютере.

Две наиболее частые причины для бесконечной рекурсии:

- 1 Неправильная запись выхода из рекурсии. Например, если мы в программе вычисления факториала забудем поставить проверку `if n == 0`, то `factorial(0)` вызовет `factorial(-1)`, тот вызовет `factorial(-2)` и т. д.
- 2 Неправильная запись параметров в функции. Например, если функция `factorial(n)` будет вызывать `factorial(n)`, то также получится бесконечная цепочка.

Поэтому при разработке рекурсивной функции в первую очередь нужно оформить условие завершения рекурсии, которое также называется **базовым условием рекурсии**.

Прямой и обратный ход рекурсии

Действия, выполняемые функцией до входа на следующий уровень рекурсии, называются выполняющимися на прямом ходу рекурсии, а действия, выполняемые по возврату с более глубокого уровня к текущему, – выполняющимися на обратном ходу рекурсии.

Задача 2. Составим процедуру, которая переводит натуральное число в двоичную систему. Стандартный алгоритм перевода числа в двоичную систему можно записать, например, так:

```
def printBin ( n ):  
    while n != 0:  
        print (n % 2, end = '')  
        n = n // 2  
printBin(14)    #для вызова функции введем для примера число 14  
>>>  
0111  #результат
```

Проблема в том, что двоичное число выводится «задом наперед», то есть первым будет выведен последний разряд.

Есть разные способы решения этой задачи, которые сводятся к тому, чтобы запоминать остатки от деления (например, в символьной строке) и затем, когда результат полностью получен, вывести его на экран.

Пробуем использовать рекурсию. Идея такова: чтобы вывести двоичную запись числа n , нужно сначала вывести двоичную запись числа $n // 2$, а затем вычислить ее остаток ($n \% 2$). Если полученное число – параметр равно нулю, нужно выйти из процедуры. Возьмем для примера число 14:

```
14 // 2 = 7, 7 % 2 = 1  
7 // 2 = 3, 3 % 2 = 1  
3 // 2 = 1, 3 % 2 = 1  
1 // 2 = 0, выход
```

Такой алгоритм очень просто программируется. Запишем функцию с назначением `print_bin`:

```
def print_bin ( n ):  
    if n == 0: return 0  
    print_bin ( n // 2 )  
    print ( n % 2, end = '')  #записывает остатки в одну  
    строку  
printBin (14)      #для вызова функции введем для примера  
число 14  
>>>  
1110  #результат
```

То же самое можно было сделать и с помощью цикла. Отсюда следует важный вывод: **рекурсия заменяет цикл**. При этом программа во многих случаях становится более понятной.

Задача 3. Запишем рекурсивную функцию для возвведения числа a в степень b ($a^{**}b$).

При возведении числа в степень, алгоритм делает следующее:

$(a * \dots (a * (a * (a * 1))))$

Из курса математики мы знаем, что $a^{**}0 = 1$.

Решение задачи:

```
def func_step(a, b):  
    if b == 0:  
        return 1  
    else:  
        return a * func_step(a, b-1)  
print(func_step(2, 4))      #например, 2 в 4 степени  
>>>  
16      #результат
```



ЗАПОМНИ

Для остановки вывода результатов бесконечной рекурсии нужно нажать на **Ctrl+C**.



ЭТО ИНТЕРЕСНО!

Примеры рекурсии в жизни человека

- Вы положили деньги в банк под проценты и начисленные проценты остаются в банке на вашем счете и на них начисляют проценты. Процесс продолжается пока вы не заберете весь вклад.
- Повторяющееся отражение в зеркалах, расположенных напротив друг друга – это бесконечная рекурсия.

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Дано натуральное число n . Выведите все числа от 1 до n .
- 2) Запишите рекурсивную функцию, которая для заданного числа n вычисляет сумму всех чисел от 1 до n .

Тема 3.2.

Алгоритмы обработки массивов

В 7 и 8 классах вы уже начали изучать массивы, представленные в виде списков, кортежей и словарей, выполнять некоторые операции с ними, вводить и выводить данные из массива.

В этой теме мы более подробно рассмотрим основные стандартные алгоритмы для обработки данных в массивах:

- поиска
- модификации
- сортировки

СМОТРИ ТАКЖЕ

Тема 3.6 8 класс

Массивы



ВСПОМНИ

Массив – это группа переменных, расположенных в памяти рядом (в соседних ячейках) и имеющих общее имя. Каждая ячейка имеет уникальный номер – индекс.



Поиск в массиве

Рассмотрим алгоритм поиска на примере задачи, в которой **a** – имя массива, **n** – количество элементов в массиве (целое число), а переменная **i** будет обозначать индекс элемента списка.

Задача 1. Необходимо найти в массиве элемент, равный значению переменной **x** или сообщить, что его нет. Алгоритм решения – это просмотр всех элементов массива с первого до последнего. Как только найден элемент, равный **x**, нужно выйти из цикла и вывести результат.

Для этого мы используем цикл с переменной **nx**, перебираем все элементы массива и досрочно завершаем цикл, если найдено требуемое значение.

```
a = [1, 5, 4, 31, 10] #Задан массив с вариантами значений
x = int(input('Введите X: ')) #Ввод искомого значения x
nx = 0 #Переменная nx сохраняет номер найденного элемента
for item in a: #для элементов массива a
    if item == x: #если элемент будет равен x
        nx = item #то оно запишется в переменную nx
        break #завершает цикл
if nx >= 1: #переменная nx - изменилась
    print ('Нашли! ') #выдает что искомое значение найдено
else:
    print ('Не нашли')
```

Для выхода из цикла используется оператор **break**, номер найденного элемента сохраняется в переменной «**nx**». Если ее значение осталось равным 0 (не изменилось в ходе выполнения цикла), то в массиве нет элемента, равного **x**.

Эту задачу можно решить еще одним способом: использовать функцию **index** для массива, которая возвращает индекс первого найденного элемента равного **x**:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18] #пример массива
print (a.index (23))
>>>
4 #результат
```

Если элемента в массиве нет, программа вернет ошибку и прекратит работу.

Задача 2. Найдем в массиве максимальный элемент, который должен записаться в переменную «**m**». Для этого надо просмотреть все элементы массива один за другим. Если очередной элемент массива больше, чем максимальный из предыдущих (находящийся в переменной «**m**»), запомним новое значение максимального элемента в «**m**».

Поскольку содержимое массива неизвестно, можно записать в «**m**» ноль или отрицательное число. Если это будет **m = a[0]**, то цикл перебора начнется со второго элемента, то есть с **a[1]**:

```

a = range (1, 20)
m = a[0]
for i in range (1, 20):
    if a[i] > m:
        m = a[i]
print (m)

```

Вот еще один вариант:

```

a = range (1, 20)
m = a[0]
for x in a:
    if x > m:
        m = x
print (m)

```

Он отличается тем, что не использует переменную-индекс, но зато дважды просматривается элемент **a[0]** (второй раз – в цикле, где выполняется перебор всех элементов).

Поскольку операции поиска максимального и минимального элементов нужны очень часто, в Python есть соответствующие встроенные функции **max()** и **min()**:

```

a = range (1,20)
m = max (a)
print (m)

```

Модификация массива

Для работы с массивами и изменения в них данных в Python есть множество различных функций, например:

ФУНКЦИЯ	ЗНАЧЕНИЕ	ПРИМЕР
print (a)	Выводит список a на экран	<pre> a = [16, 'b', 34, 'c']#базовый список для всех примеров print (a) >>> [16, 'b', 34, 'c'] </pre>
append ()	Добавляет один элемент в конец списка	<pre> a.append (18) print (a) >>> [16, 'b', 34, 'c', 18] </pre>
clear ()	Удаляет все элементы списка	<pre> a.clear () print (a) >>> [] </pre>

ФУНКЦИЯ	ЗНАЧЕНИЕ	ПРИМЕР
count ()	Возвращает количество элементов с заданным значением	<pre>print (a.count (16)) #считает сколько элементов со значением 16 есть в списке >>> 1</pre>
extend ()	Добавляет другой список в конец базового списка	<pre>B = [18, 'h'] #второй список a.extend (b) print (a) >>> [16, 'b', 34, 'c', 18, 'h']</pre>
index ()	Возвращает номер индекса первого найденного похожего элемента	<pre>print (a.index (34)) >>> 2</pre>
insert ()	Вставляет элемент по индексу	<pre>a.insert (1, 22) #поскольку отсчет индекса идет с 0, то в базовом списке под индексом 1 стоит элемент 'b'; вместо него должна вставиться цифра 22, остальные сдвигаются, вправо. print (a) >>> [16, 22, 'b', 34, 'c']</pre>
pop ()	Удаляет элемент под заданным индексом	<pre>a.pop (0) #удалить значение под индексом 0 print (a) >>> ['b', 34, 'c'] a.pop () print (a) >>> [16, 'b', 34] #если значение не задаются, то удаляется последний элемент</pre>
remove ()	Удаляет первый элемент с заданным значением. Если элемент не найден – то выдается <code>ValueError</code>	<pre>a.remove (34) print (a) >>> [16, 'b', 'c']</pre>
reverse ()	Переставляет элементы списка в обратном порядке	<pre>a.reverse () print (a) >>> ['c', 34, 'b', 16]</pre>
sort ()	Сортирует список (только для списков с одним типом элементов)	<pre>a = [16, 8, 34, 3] #в списке только числа(int) a.sort () print (a) >>> [3, 8, 16, 34] #сортирует по возрастанию a = ['m', 'b', 'o', 'c'] #в списке только символы (str) a.sort () print (a) #сортирует по алфавиту >>> ['b', 'c', 'm', 'o']</pre>

Рассмотрим более подробно некоторые из них: реверс массива, сдвиг элементов массива и отбор нужных элементов.

Реверс массива

Реверс массива (*reverse*) – это перестановка его элементов в обратном порядке: первый элемент становится последним, а последний – первым.

Индекс элементов массива в Python начинаются с 0. Поэтому, если общее количество элементов равно N , то противоположный для i элемент находится по формуле: $N - i - 1$. Например (см.таблицу ниже), чтобы найти индекс элемента, который заменит элемент под индексом 2, вычисляем по формуле:

$$\begin{array}{ll} N & i \\ 9 & -2 - 1 = 6 \end{array}$$

Элемент под индексом 2 должен поменяться местами с элементом под индексом 6.

Элементы массива	16	29	-5	-11	23	14	-7	23	18
Индексы	0	1	2	3	4	5	6	7 или ($n-2$)	8 или ($n-1$)

Чтобы найти вторую пару элемента, мы просматриваем индексы только первой половины массива. Это значит, что цикл нужно остановить на середине массива:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
n = len(a)      #вычисляем количество элементов в массиве
for i in range(n//2):#только индексы первой половины массива
    a[i], a[n-i-1] = a[n-i-1],
a[i]  #переставляем элементы
print(a)
>>>
[18, 23, -7, 14, 23, -11, -5,
29, 16]
```

Операция реверс массива может быть выполнена и с помощью стандартного метода **reverse()**:

```
a.reverse()
```

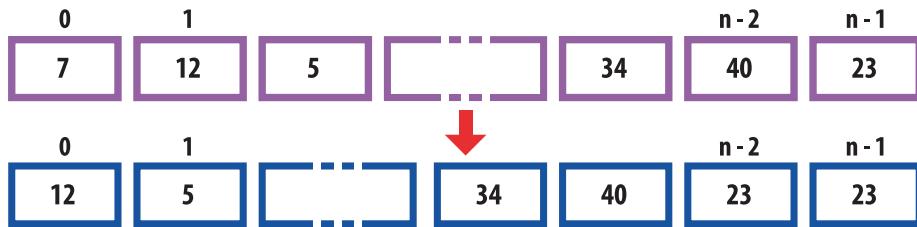


ЗАПОМНИ

Значения индексов в Python могут быть отрицательными. В этом случае нумерация будет идти с конца. Например, индекс самого последнего элемента массива равен -1 , предпоследнего -2 и т.д.

Сдвиг элементов массива

При удалении и вставке элементов необходимо выполнять сдвиг части или всех элементов массива в ту или другую сторону. Массив часто рисуют в виде таблицы, где первый элемент расположен слева. Поэтому сдвиг влево – это перемещение всех элементов на одну ячейку, при котором $a[1]$ переходит на место $a[0]$, $a[2]$ – на место $a[1]$ и т.д.



Последний элемент остается на своем месте, то есть дублируется, поскольку новое значение для него взять неоткуда – массив кончился.

Запишем алгоритм:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
n = len(a)
for i in range (n-1):
    a[i] = a[i+1]
print(a)
>>>
[29, -5, -11, 23, 14, -7, 23, 18]
```

Обратите внимание, что цикл заканчивается при $a = [n-1]$, чтобы не было выхода за границы массива, то есть обращения к несуществующему элементу $a[n]$.

Как видим, первый элемент пропал, а последний – повторился 2 раза. Можно старое значение первого элемента записать на место последнего. Такой сдвиг называется циклическим. Для этого предварительно (до начала цикла) первый элемент нужно запомнить во вспомогательной переменной (c), а после завершения цикла записать его в последнюю ячейку массива:

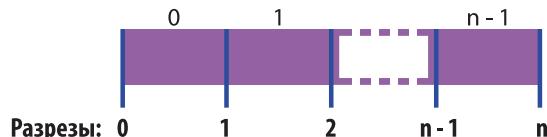
```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
n = len(a)
c = a[0]
for i in range(n-1):
    a[i] = a[i+1]
a[n-1] = c
print(a)
>>>
[29, -5, -11, 23, 14, -7, 23, 18, 16]
```

Можно выполнить сдвиг, используя встроенные возможности Python:

```
a = a[1:n] + [a[0]]
```

Здесь первый элемент $a[0]$ ставится в конец к обрезанному массиву $a[1:n]$, который теперь начинается не с нуля, а с единицы.

На рисунке справа мы видим, что срез $a[0:2]$ включает все элементы между 0 и 2, то есть элементы $a[0]$ и $a[1]$.



Отбор нужных элементов

Требуется отобрать все элементы массива **a**, удовлетворяющие некоторому условию, в новый массив **b**. Для этого перебираем все элементы исходного массива и, если очередной элемент нам подходит, добавляем его в новый массив, используя второй счетчик. Н-р, соберем во второй список четные числа:

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
b = []
for x in a:
    if x % 2 == 0:
        b.append(x)
print(b)
>>>
[16, 14, 18]
```

Второй вариант решения – использование генератора с условием

```
a = [16, 29, -5, -11, 23, 14, -7, 23, 18]
b = [x for x in a if x % 2 == 0]
print(b)
```

Здесь в новый список **b** отбираются только элементы, делящиеся на 2.

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Заполните массив случайными числами в интервале (0,20). Введите число x и найдите все значения, равные x .
- 2) Дан одномерный массив числовых значений, насчитывающий n элементов. Выполните перемещение элементов массива по кругу вправо, т.е. $a(1) \rightarrow a(2); a(2) \rightarrow a(3); \dots a(n) \rightarrow a(1)$.

Тема 3.3.

Сортировка списков

Часто для того чтобы облегчить поиск нужной информации, мы пользуемся сортировкой. Например, сортировка слов по алфавиту облегчает поиск слова в словаре.

В программировании **сортировка** – это перестановка элементов массива в заданном порядке.

Порядок сортировки может быть любым: для чисел обычно рассматривают сортировку по возрастанию (или убыванию) значений. Например, при сортировке по возрастанию из одномерного массива [3 1 0 5 2 7] получается массив [0 1 2 3 5 7]. Символьные данные обычно сортируются в алфавитном порядке.

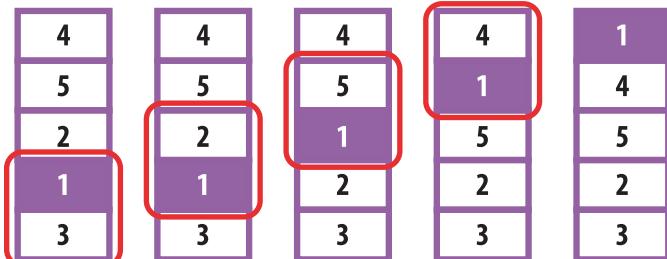
Было придумано множество способов сортировки. В целом их можно разделить на две группы: 1) простые, но медленно работающие (на больших массивах) и 2) сложные, но быстрые.

Рассмотрим один из наиболее наглядных методов сортировки – «метод пузырька».

Метод пузырька (сортировка обменами)

Название этого метода произошло от известного физического явления – пузырек воздуха в воде поднимается вверх. Для сортировки массива таким способом необходимо справа налево пройтись по массиву, попарно сравнивая соседние элементы, и в том случае, когда левый больше правого, менять их местами. Так самые «тяжелые» элементы падают на дно, а самые «легкие» (минимальные) элементы поднимаются наверх (к началу массива) подобно пузырькам воздуха.

Далее так же рассматриваем следующую пару элементов от конца и т.д. (см. рисунок).



Когда мы обработали пару ($a[0]$, $a[1]$), минимальный элемент стоит на месте $a[0]$. Это значит, что на следующих этапах его можно не рассматривать. Первый цикл, устанавливающий на свое место первый (минимальный) элемент, можно на псевдокоде записать так:

```
для i от n-2 до 0 шаг -1
    if (a[j] > a[j+1])
        поменять местами a[j] и a[j+1]
```

Переменная i хранит индекс ячейки, в которую записывается минимальный элемент. Сначала это будет первая ячейка. Переменная j используется для обращения к текущим сравниваемым ячейкам массива.

За один проход такой цикл ставит на место один элемент. Чтобы «подтянуть» второй элемент, нужно написать еще один почти такой же цикл, который будет отличаться только конечным значением i в заголовке цикла. Так как верхний элемент уже стоит на месте, его не нужно трогать:

```
for i in range(n-1): #количество переходов
    for j in range(n-i-1):
        if a[j] > a[j+1]:
            a[j], a[j+1] = a[j+1], a[j]
```

Таких циклов нужно сделать $n-1$ – на 1 меньше, чем количество элементов массива. Почему не n ? Дело в том, что если $n-1$ элементов поставлены на свои места, то оставшийся автоматически встает на свое место – другого места нет.

Запишем полную программу сортировки методом пузырька:

```
from random import randint
n = 10
a = [randint (1,99) for n in range (n)]
print (a)
for i in range (n-1): #кол-во проходов по списку
    for j in range (n-i-1): #кол-во сравнений уменьшается на величину i
        if a [j] > a [j+1]: #вложенный цикл сравнивает эл-т j с j+1
            a[j], a [j+1] = a[j+1], a[j] #при необходимости
меняет местами
print(a)
>>>
[5, 84, 90, 37, 30, 32, 29, 62, 17, 99]
[5, 17, 29, 30, 32, 37, 62, 84, 90, 99]
```

Метод выбора

Еще один популярный и простой метод сортировки – метод выбора. Сначала просматривается весь массив, находится минимальный элемент и переставляется в самое начало массива. Во втором проходе, среди всех оставшихся, то есть со второго по последний элемент, снова находится наименьший элемент и переставляется местом со 2-м. Далее среди элементов, начиная с 3-го, также находится наименьший и меняется с 3-м и так далее до $(n-1)$ -го элемента.

Запишем алгоритм для сортировки списка методом выбора, где переменная i хранит индекс ячейки, в которую записывается минимальный элемент, а переменная j – просматриваемый элемент:

```
for i in range(n-1):
    n_min = i
    for j in range(i+1, n):
        if a[j] < a[n_min]:
            n_min = j
    if i != n_min:
        a[i], a[n_min] = a[n_min], a[i]
```

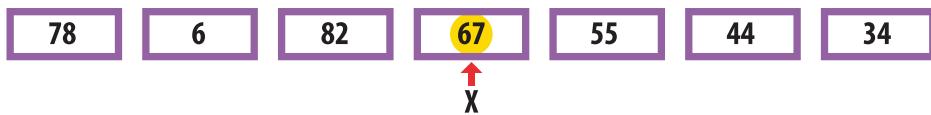
Здесь перестановка происходит только тогда, когда найденный минимальный элемент стоит не на своем месте, то есть $i \neq n_min$. Поскольку поиск номера минимального элемента выполняется в цикле, этот алгоритм сортировки также представляет собой вложенный цикл.

«Быстрая сортировка» или quicksort

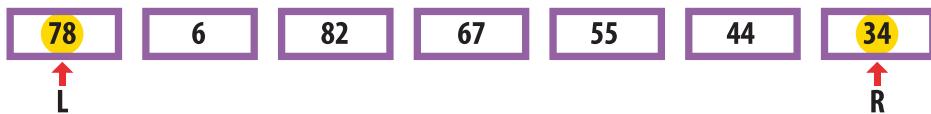
Пузырьковая сортировка и метод выбора работают медленно для больших массивов данных (более 10000 элементов). Поэтому для сортировки больших массивов часто используется рекурсивный алгоритм «быстрой сортировки» (англ. quicksort).

Идея такой сортировки состоит в том, что сначала массив разбивается где-то посередине, затем из элементов левой части выбираются те, которые больше или равны значению «среднего элемента» и переставляются в правую часть. Далее левая и правая части рассматриваются как отдельные массивы и снова делятся на два.

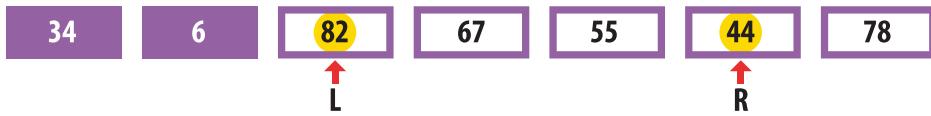
Чтобы понять сущность метода, рассмотрим пример. Пусть задан массив:



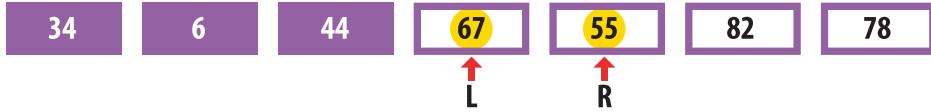
Выберем в качестве **X** средний элемент массива, то есть 67. Найдем первый слева элемент массива, который больше или равен **X** и должен стоять во второй части. Это число 78. Обозначим индекс этого элемента через **L**. Теперь находим самый правый элемент, который меньше **X** и должен стоять в первой части. Это число 34. Обозначим его индекс через **R**.



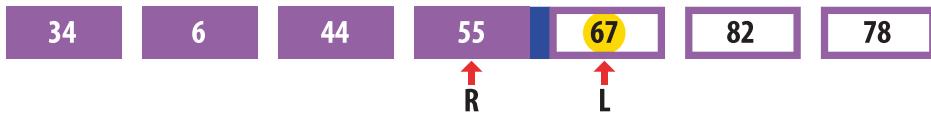
Теперь поменяем местами два этих элемента. Сдвигая переменную **L** вправо, а **R** – влево, находим следующую пару, которую надо переставить. Это числа 82 и 44.



Следующая пара элементов для перестановки – числа 67 и 55.



После этой перестановки дальнейший поиск приводит к тому, что переменная **L** становится больше **R**, то есть массив разбит на две части. В результате все элементы массива, расположенные левее **A[L]**, меньше или равны **X**, а все правее **A[R]** – больше или равны **X**.



Таким образом, сортировка исходного массива свелась к двум сортировкам частей массива, то есть к двум задачам того же типа, но меньшего размера. Теперь нужно применить тот же алгоритм к двум полученным частям массива: первая часть – с 1-го до **R**-го элемента, вторая часть – с **L**-го до последнего элемента.

Как мы уже говорили, скорость сортировки важна при работе с большими массивами. Ниже в таблице сравнивается время сортировки (в секундах)

массивов разного размера, заполненных случайными значениями, с использованием трех изученных алгоритмов.

N	Метод пузырька	Метод выбора	Быстрая сортировка
1000	0,09 с	0,05 с	0,002 с
5000	2,4 с	1,2 с	0,014 с
15000	22 с	11 с	0,046 с

Как видно из таблицы, при обработке массива, к примеру, с 15 тысячами элементов, быстрая сортировка работает в 500 раз быстрее, чем при методе пузырька.

Из предыдущей темы мы также знаем, что в Python есть встроенная функция для сортировки массивов **sorted**, которая использует гибридный алгоритм **timesort** и легко справляется с обработкой большинства известных данных. Необходимо также помнить, что при записи:

```
a.sort ()      #сортируется сам список a.  
b = sorted (a) #в массив b переносится отсортированный в по-  
рядке возрастания массив a.
```

По умолчанию сортировка выполняется по возрастанию или точнее «неубыванию», когда каждый следующий элемент больше или равен предыдущему. Для того чтобы отсортировать массив по убыванию (невозрастанию – каждый следующий элемент меньше или равен предыдущему), нужно в функции сортировки указать `reverse = True`:

```
b = sorted (a, reverse = True) #или  
a.sort (reverse = True)
```

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Дан одномерный массив числовых значений, насчитывающий n элементов. Из элементов исходного массива построить два новых. В первый должны входить только числа, которые делятся на 3, а во второй – числа, которые делятся на 5.
- 2) Продолжите программу из первого вопроса и допишите алгоритм, который сортирует числа, делящиеся на 3 по возрастанию, а все числа делящиеся на 5 – по убыванию.

Тема 3.4.

Матрицы

Матрица – это двумерный массив, имеющий табличную структуру. Описываются массивы так же, как одномерные. Разница состоит в том, что у элемента двумерного массива две координаты (два индекса) – номер строки и номер столбца, в которых находится элемент.

Например, при составлении программы для игры в крестики-нолики можно пустым клеткам присвоить код «-1», клетке с ноликом – код 0, а клетке с крестиком – код 1:

	0	1	2
0	-1	0	1
1	-1	0	1
2	0	1	-1

В Python для работы с таблицами используют списки. Двумерная таблица хранится как список, каждый элемент которого тоже представляет собой список («список списков»). Например, таблицу, показанную на рисунке, можно записать так:

```
a = [[-1, 0, 1],
      [-1, 0, 1],
      [0, 1, -1]]
```

Данный оператор можно записать в одну строку:

```
a = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]]
```

Но поскольку человек воспринимает матрицу как таблицу, лучше и на экран выводить ее в виде таблицы. Для этого матрицу можно записать так:

```
a = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]]
for i in range ( len(a) ):
    for j in range ( len(a[i]) ):
        print ( '{:4d}'.format(a[i][j]), end = ' ' )
    print ()
>>>
```

Формат вывода `{:4d}` – раздвигает поля между столбцами: в нашем примере перед каждым элементом ставится 4 пробела.

```
-1      0      1
-1      0      1
 0      1     -1
```

Здесь **i** – индекс подсписка (определяет число строк), а **j** – индекс элемента внутри подсписка (определяет число столбцов); **len(a)** – это число подсписков в большом списке (их здесь 3), **len(a[i])** – число элементов подсписка, которое совпадает с числом столбцов.

a [i] [j] – это элемент в подсписке **i**, под **j**-индексом:

a [0] [0]==-1, a [0] [1]==0, a [0] [2]==1, a [1] [0]==-1, и т.д.

Этот же пример можно записать так:

```
for row in a:          #в строке a
    for elem in row:    #для элемента в строке
        print(elem, end=' ') #вывести элементы
    print()
```

Задача 1. Заполним матрицу случайными цифрами. Количество строк и столбцов введем с клавиатуры.

Каждому элементу матрицы можно присвоить любое значение. Поскольку индексов два, для заполнения матрицы нужно использовать вложенный цикл. Далее будем считать, что существует матрица **a**, состоящая из **n** строк и **m** столбцов, а **i** и **j** – целочисленные переменные, обозначающие индексы строки и столбца. В этом примере матрица заполняется случайными числами и выводится на экран:

```
import random
n = int(input ('Введите количество строк: '))
m = int(input ('Введите количество столбцов: '))
a = []
for i in range (n):
    a.append([])
    for j in range (m):
        a[i].append (random.randint (10,40)) #каждому э-ту
присваивается случайное число от 10 до 40
for i in a:      #каждый подсписок выводится с новой строки,
    print (i)    #но в квадратных скобках
>>>
[33, 16, 31, 33]      #случайные числа при n=2, m=4
[39, 35, 11, 15]
```

Эту же задачу запишем короче, и так, чтобы в результате не было скобок:

```
import random
n = int(input ('Введите количество строк: '))
m = int(input ('Введите количество столбцов: '))
a = [[random.randint(10, 40) for i in range(m)] for j in
range(n)]
print(' '.join([str(elem) for elem in row])) #все элементы
объединяются и перечисляются через пробел
```

Обработка двумерного массива

Такой же двойной цикл нужно использовать для перебора всех элементов матрицы. Первый цикл перебирает номер строки, второй цикл бежит по элементам внутри строки. Вот как вычисляется сумма (s) всех элементов:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print(s) #результат 45
```

Для этой записи можно использовать встроенную функцию **sum**:

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    s += sum(row)
print (s)
```

Обработку некоторых элементов матрицы рассмотрим на примере задачи:

Задача 2. Пусть дан квадратный массив из n строк и n столбцов. Необходимо заполнить диагональ единичками, область слева от нее заполнить двойками, а область справа – нулями.

```
1 0 0 0 Главная диагональ – это элементы a[0, 0], a[1, 1], ..., a[n-1, n-1],
2 1 0 0 то есть номер строки равен номеру столбца. Для заполнения
2 2 1 0 главной диагонали единицами нужен один цикл:
2 2 2 1 for i in range(n):      #работаем с a[i][i]
              a[i][i] = 1          #заполняем их единицами
```

Элементы справа от диагонали заполним значением 0, для чего нам понадобится в каждой из строк с номером i присвоить значение элементам $a[i][j]$ для $j=i+1, \dots, n-1$. Здесь нам понадобятся вложенные циклы:

```
for i in range(n):
    for j in range(i + 1, n):
        a[i][j] = 0
```

Аналогично присваиваем значение 2 элементам $a[i][j]$ для $j=0, \dots, i-1$:

```
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
```

Если внешние циклы объединить в один, то можно получить одно такое решение:

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
    a[i][i] = 1
    for j in range(i + 1, n):
        a[i][j] = 0
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ:

- 1) Заполните прямоугольную матрицу a , имеющую n строк и m столбцов элементами из случайных чисел. Найдите среднее арифметическое элементов массива.
- 2) Найдите наибольшее значение среди средних значений для каждой строки матрицы.

Глава

Ч



Компьютерные сети и интернет

Тема 4.1.

Технологии будущего

Развитие интернет-технологий привело к тому, что мир необратимо изменился. Компьютеры и девайсы стали неотъемлемой частью картины настоящего, и мы все понимаем, что именно эти устройства будут повсеместно в будущем.



ЭТО ИНТЕРЕСНО!

Футуролог (предсказатель будущего), изобретатель, исследователь развития технологий и технический директор Google в области машинного обучения **Рэй Курцвейл** каждые несколько лет публикует технологические прогнозы, часть из которых уже сбылась. Например, он предсказал победу компьютера в шахматной партии с лучшим шахматистом мира, появление систем виртуальной и дополненной реальности.



В этой теме мы хотели бы рассказать немного о так называемых «прорывных технологиях» (англ. *disruptive technologies*), широкое внедрение которых полностью изменит наше представление о мире в следующие несколько десятилетий.

1 Искусственный интеллект

Искусственный интеллект (ИИ; англ. *artificial intelligence, AI*) – это особый тип интеллекта, присущий компьютерам и машинам, которые копируют когнитивные функции человеческого мозга. ИИ нацелен на то, чтобы понять, как человек принимает те или иные решения.

Главное превосходство интеллектуальных компьютерных программ над другими программами – это их способность самообучаться и исправлять ошибки внутри себя. Чем больше ИИ взаимодействует с человеком и с другими программами, тем больше новой информации он запоминает и она становится частью их «жизненного опыта».

Технологии ИИ интенсивно развиваются благодаря широкому распространению нейронных сетей и облачным вычислениям. По прогнозам Курцвейла, уже к 2030 году ИИ будет способен мыслить полностью как человек.

2 Интернет вещей (англ. *internet of things, IoT*)

Компьютеры продолжают уменьшаться, и теперь они настолько малы, что их можно вшивать в одежду, вставлять в зубные щетки, часы и лампы. Все это позволяет нам управлять всеми нашими вещами удаленно – через интернет. Теперь каждый девайс и одежда имеют свой IP-адрес в сети, через которую они взаимодействуют между собой.



«Умные приборы», следящие за нашим здоровьем, могут предупреждать нас о приближающейся простуде. Каждый троллейбус и автобус, оснащенный датчиком, теперь можно увидеть на интернет-карте города. «Умные» уличные фонари отключаются, как только уровень света опускается ниже допустимого, а светофоры светят зеленым дольше, если видят, что поток машин с одной стороны больше, чем с другой.

«Умнеют» не только отдельные машины, но и целые города и страны. Вы уже, наверное, видели в фильмах, как полицейские нажатием одной кнопки получают картинку с любой камеры видеонаблюдения страны. Несмотря на все преимущества, прежде чем подключить каждый свой девайс к интернету, задумайтесь, а хотите ли вы оставлять везде за собой такой «цифровой» след?

3 Робототехника

Роботы уже давно среди нас. И хотя они еще выглядят не как полная копия человека, оснащенные искусственным интеллектом, они уже принимают на себя роли, которые когда-то могли выполнять только люди. По прогнозам футурологов, роботы станут для нас такой же привычной вещью в доме, как холодильник, уже в ближайшие 10 лет.

Роботы становятся частью самого человека. К примеру, через вживление роботизированных или так называемых «бионических протезов». Сейчас для управления таким протезом используются системы, считывающие сигналы с близлежащих мышц.



Однако не за горами и «вживление» протеза, когда владелец такого протеза получает обратную информацию о качествах объекта, к которому он прикасается устройством (горячий, шершавый и т.д.). Самые простые протезы уже доступны, и их можно легко напечатать на 3D-принтере.

4 3D-печать

Что произойдет, если мы сможем создать что-либо, в том числе автомобили, просто печатая их? Хотя это трудно представить, но это уже происходит: люди печатают себе дома, мебель, посуду, транспорт, одежду, игры, части тела. Даже можно печатать еду, потому что появились кондитерские 3D-принтеры! Вы можете себе представить, что вы напечатаете себе яблоко, а оно будет кислым, потому что в картридже у вас закончился сахар?

С широким развитием 3D-печати уже не нужно будет перевозить огромные контейнеры с товарами через полмира. Будет достаточно того, что вам отправят файл с чертежами нужного вам товара. Возможно, благодаря этому, мы сможем остановить вредные производства, а также загрязнение воздуха, вызванное глобальной транспортной системой.

5 Биотехнологии и генная инженерия

На момент написания учебника уже проходят тысячи медицинских испытаний по разработке различных прорывных биотехнологий. Разрабатываются новые вакцины, проводятся клинические испытания по редактированию генов с использованием технологии CRISPR, когда проводится искусственное и целенаправленное изменение генотипа биологических организмов. Основной метод генной инженерии – выделение необходимых генов и их модификация: исправление мутированных генов, восстановление отсутствующих генов и т.д. Это особенно актуально при лечении болезней, связанных с нарушениями в иммунной системе, в системе свертывания крови, в онкологии.

Другое стремительно развивающееся направление – это наномедицина, которая начинается с разработки наноматериалов и заканчивается наноэлектронными биосенсорами. Например, разрабатываются такие нанороботы, которые могут проникать в клетки организма, чтобы «кормить» их и устранять отходы.



6 Виртуальная реальность (VR)

К концу 2030 года VR будет такого высокого качества, что ее будет невозможно отличить от настоящей реальности. Развитие сети интернет позволяет создать виртуальный мир, которым пользуются сразу несколько пользователей.

Сегодня VR активно применяется не только в индустрии развлечений, но и в автомобилестроении, аэрокосмической промышленности и судостроении. Технология VR позволяет сокращать время разработки и проводить испытания продуктов, которые еще не существуют.

7 Возобновляемая энергия и зеленые технологии

Возобновляемая или «зеленая» энергия – это энергия из источников, которые, по человеческим масштабам, являются неисчерпаемыми. По мнению экспертов, возобновляемые источники энергии могут минимизировать изменение климата и загрязнение нашей планеты.

Возобновляемая энергия становится реальностью благодаря таким технологическим изобретениям, как ветряные турбины, фотоэлектрические элементы, солнечные панели, геотермальная энергия, энергия океанских волн и другим.

Также увеличение мощностей накопителей энергии при сохранении своих габаритов – серьезно повышает спрос на возобновляемые источники энергии.



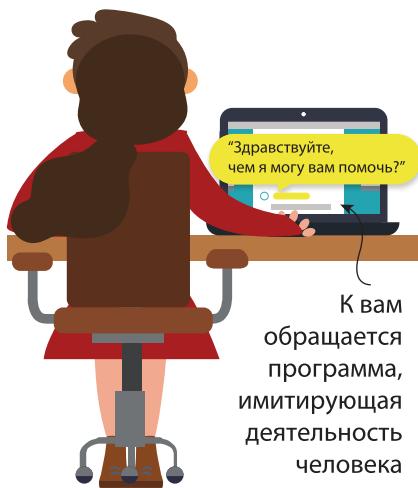
ВОПРОСЫ И ЗАДАНИЯ:

- 1) Какие возобновляемые источники энергии вы знаете? Почему, по вашему мнению, использование «зеленых» технологий важно для нашей планеты?
- 2) Считаете ли вы этичным генное модифицирование?
- 3) Видите ли вы угрозу в использовании 3D-принтеров, ИИ, виртуальной реальности? Чем эти технологии могут быть опасны для человечества?

Тема 4.2.

Безопасность в цифровом мире

Цифровой мир или как его еще называют – киберпространство – это мир компьютеров и компьютерных сетей, параллельная вселенная со своими законами и правилами; пространство, открывающее безграничные возможности и таящее опасности и угрозы.



Чтобы избежать неприятностей в виртуальном пространстве, необходимо соблюдать некоторые правила.

Первое, что мы должны помнить, входя в интернет – это открытый и многоголосый мир и мы в нем не одни. За каждым профилем, аккаунтом находится живой человек, группа людей или программный бот.

Работая в интернете, нужно помнить о том, что не только боты могут выдавать себя за людей. Люди в сети тоже могут выдавать себя за людей – за других людей.

Такие люди могут преследовать самые различные цели – от взлома вашего аккаунта (кража вашего пароля, использование вашего профиля в различных целях) до физического преследования, шантажа и вовлечения в опасные взаимодействия.

Помните о том, что вся информация, которой вы делитесь в интернете, может быть использована злоумышленниками. Думайте, о чем говорить в сети, какие данные размещать, кого принимать в друзья.

ОПРЕДЕЛЕНИЯ

«Бот» (сокращение от «робот») – программа, имитирующая деятельность человека. Чат-бот имитирует собеседника в чате.



Как мы уже сказали, в интернете существует много возможностей, чтобы выдавать себя за другого. У одного и того же пользователя может быть сразу несколько аккаунтов, где он и подросток, и военнослужащий, и государственный чиновник.

Если вам пришло неожиданное сообщение от друга, в котором он просит вас забросить ему деньги на телефон, прийти на встречу в назначенное место или совершить еще какие-то действия, попробуйте связаться с другом каким-то иным способом и уточнить, действительно ли сообщение отправил он. Возможно, его аккаунт был взломан и от его имени вам писали незнакомые люди.

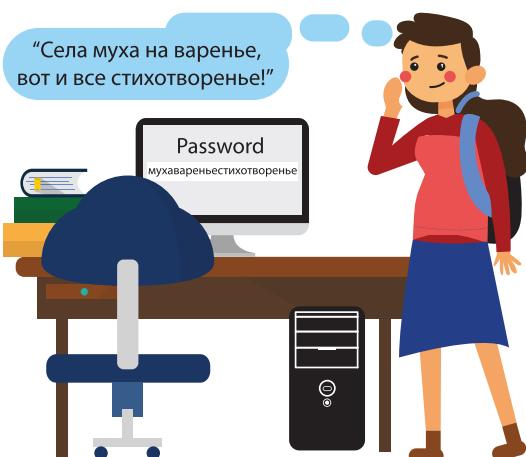
Также к вам могут приходить сообщения и электронные письма с ссылками, по которым вас просят пройти или вложения (фото, видео, документы), которые вам предлагают открыть. Предложения могут быть завлекательным (вы выиграли миллион!) или пугающими (мы расскажем всем твою тайну!). Такие сообщения чаще всего имеют единственную цель – заразить ваше устройство вирусом или похитить ваш аккаунт. Не идите на поводу у злоумышленников!

Наши аккаунты – это информация о нас, наших близких, наших отношениях. Оказавшись в чужих руках, они могут нанести вред не только нам, но и всем, кто с нами контактирует. Чтобы этого не случилось, аккаунт нужно надежно защитить. Одна из основных мер защиты аккаунта – надежный пароль.

Надежный пароль содержит не менее двенадцати символов, включая прописные и строчные буквы, цифры и специальные символы (знак решетки, звездочка, нижнее подчеркивание и т.д.).

В надежном пароле не используется личная информация (имя, фамилия, день рождения, кличка домашнего животного, увлечение и то, что можно узнать о вас из социальных сетей).

Надежный пароль запомнить нелегко. Для этого используются различные технологии запоминания. Например, вы можете объединить три-четыре слова, связанные логической цепью, понятной только вам.



Еще лучше будет разделить объединенные слова цифрами или символами.

К примеру: *муха1варенье2стихотворенье3*

И совсем замечательно: *muh@1v@renie2stihotvorenie3*

Никогда не используйте пароль, предложенный в учебнике! Придумайте свой!

Никогда не используйте один и тот же пароль на разных сайтах!

Для того чтобы не мучиться с запоминанием различных паролей, можно использовать специальные программы – парольные менеджеры.

Парольные менеджеры помогают создавать, хранить и использовать пароли. Среди самых популярных таких программ можно выделить KeePass и LastPass.



Даже если ваш пароль очень надежный, все равно существует вероятность того, что он может оказаться в чужих руках. Например, если кто-то подсмотрит как вы его набираете или получит его с помощью фишинга.

Для обеспечения своей безопасности нужно внимательно читать условия предоставления каких-либо новых услуг или подписок, которые вы оформляете в интернете. Например, подписка на различные услуги типа «анекдоты» или «новый рингтон», которые предлагаются на сайтах мобильных операторов. Часто вы видите большую привлекательную кнопку «Читать», и уже одним нажатием на нее подписываетесь на услуги, за которые с вас будут брать абонентскую плату.

Чтобы защититься, всегда читайте условия (они обычно написаны мелким шрифтом) и не игнорируйте СМС-уведомления об оформлении подозрительных услуг от оператора. Если вы не можете самостоятельно справиться с отпиской в мобильном личном кабинете, обратитесь в салон связи своего оператора с просьбой отключить услугу и по возможности заблокировать возможность оформления новых подписок.



Фишинг (*“fishing”* «рыбная ловля, выуживание») – это сетевое мошенничество, направленное на получение доступа к чужим данным и аккаунтам. Часто это делается путем подсывания пользователю поддельного сайта, внешне не отличимого от настоящего, где пользователям предлагают ввести логин и пароль, после чего пароль попадает в руки мошенников, и пользователь теряет доступ к своему аккаунту.

Чтобы защитить пользователей от угроз потери аккаунта, современные сервисы предлагают использовать двухфакторную аутентификацию.

Двухфакторная аутентификация – это дополнительный уровень защиты учетной записи, который подтверждает, что войти в аккаунт намеревается его истинный хозяин, как правило, делается это с помощью телефона, указанного при включении двухфакторной аутентификации.

Для того чтобы войти в свой аккаунт, пользователю с двухфакторной аутентификацией необходимо помимо пароля ввести код, который приходит на телефон в СМС-сообщении или генерируется на телефоне с помощью специального приложения.

Таким образом, злоумышленник, даже имея пароль от вашего аккаунта, не сможет им воспользоваться, не имея доступа к вашему телефону.

Не забывайте ставить пароль на свой телефон!

Соблюдение этих правил позволит вам чувствовать себя увереннее и защищать свою информацию от несанкционированного доступа.

Все вышесказанное – это только часть рекомендаций по обеспечению безопасности в цифровом пространстве. Туда же входят антивирусная защита, создание резервных копий информации и много всего другого.

И помните главное правило безопасности – думающего человека обмануть сложнее.

ВОПРОСЫ И ЗАДАНИЯ:

- 1) Подумайте на тему того, каким образом вирусы могут попасть на ваш компьютер. Как от этого защититься?
- 2) Какие угрозы вашей цифровой информации (кроме вирусов) вы можете назвать.
- 3) Какую информацию о себе и своих близких не стоит публиковать в интернете и почему?

<https://new.bizdin.kg/>

Приложения

Приложение №1

Стандартные кодировки текста СР-1251:

Á	à	,	è	"	...	†	‡	€	%	É	<	и	и	Ó	Ý
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
á	'	,	"	"	•	-	—	ë	™	é	>	ò	и	ó	ú
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
nbsp	ÿ	ы	Э	¤	ы		§	Ё	©	Ю	«	¬	shy	®	Я
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
°	±	ы	э	'	μ	¶	•	ë	№	ю	»	э	ю	я	я
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	ы	ь	Э	Ю	Я
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Стандартные кодировки текста КОИ-8:

-		Г	Г	Л	Л	Г	Г	Т	+	+	+	+	140	141	142	143
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	
ÿ				—	•	¬	≈	≤	≥	nbsp	J	°	2	•	÷	
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	
=		F	ё	∏	∏	∏	∏	∏	∏	∏	∏	∏	∏	∏	∏	
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	
		‡	Ё		‡											
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	
ю	а	б	ц	д	е	ф	г	х	и	й	к	л	м	н	о	
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	
п	я	р	с	т	у	ж	в	ь	ы	з	ш	э	щ	ч	ъ	
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	
Ю	А	Б	Ц	Д	Е	Ф	Г	Х	И	Й	К	Л	М	Н	О	
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	
П	Я	Р	С	Т	У	Ж	В	Ь	Ы	З	Ш	Э	Щ	Ч	Ъ	
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	

Приложение №2

Элемент электронной таблицы	Способ выделения
Ячейка	Установить курсор в ячейку и нажать ЛКМ, при этом изменится цвет имени столбца и номера строки, расположенные на линейках (ячейка становится активной).
Диапазон смежных ячеек	<p>1 способ: установить курсор в первую угловую ячейку диапазона, удерживая нажатой клавишу Shift, переместить курсор в противоположный угол диапазона, щелкнуть ЛКМ.</p> <p>2 способ: установить курсор в первую угловую ячейку, удерживая ЛКМ, провести по диагонали в противоположный угол диапазона.</p>
Диапазон несмежных ячеек	Удерживая клавишу Ctrl, выделить отдельные диапазоны, используя способы выделения смежных ячеек.
Строка	Щелкнуть по номеру строки на левой линейке.
Столбец	Щелкнуть по имени столбца на верхней линейке.
Лист	Щелкнуть по прямоугольнику, расположенному между заголовком столбца A и заголовком строки 1.

Приложение №3

Арифметические операторы		
Оператор	Значение	Примеры
+	Сложение - суммирует значения слева и справа от оператора	$10 + 5$ в результате будет 15 $27 - 3$ в результате будет 24 $9.4 + 7$ в результате будет 16.4
-	Вычитание - вычитает правый операнд из левого	$15 - 5$ в результате будет 10 $20 - 3$ в результате будет 23 $13.4 - 7$ в результате будет 6.4
*	Умножение - перемножает операнды	$5 * 5$ в результате будет 25 $7 * 3.2$ в результате будет 22.4 $-3 * 12$ в результате будет -36
/	Деление - делит левый операнд на правый	$15 / 5$ в результате будет 3 $5 / 2$ в результате будет 2.5
%	Деление по модулю - делит левый операнд на правый и возвращает остаток.	$6 \% 2$ в результате будет 0 $7 \% 2$ в результате будет 1 $13.2 \% 5$ в результате будет 3.19999999
**	Возведение в степень - возведет левый операнд в степень правого	$5 ** 2$ в результате будет 25 $2 ** 3$ в результате будет 8 $-3 ** 2$ в результате будет -9
//	Целочисленное деление – деление, в котором возвращается только целая часть результата. Часть после запятой отбрасывается.	$12 // 5$ в результате будет 2 $4 // 3$ в результате будет 1 $25 // 6$ в результате будет 4

Операторы сравнения

==	Проверяет, равны ли оба операнда. Если да, то условие становится истинным.	<i>5 == 5 в результате будет True True == False в результате будет False "hello" == "hello" в результате будет True</i>
!=	Проверяет, равны ли оба операнда. Если нет, то условие становится истинным.	<i>12 != 5 в результате будет True False != False в результате будет False "hi" != "Hi" в результате будет True</i>
<>	Проверяет, равны ли оба операнда. Если нет, то условие становится истинным.	<i>12 <> 5 в результате будет True. Похоже на оператор !=</i>
>	Проверяет, больше ли значение левого операнда чем значение правого. Если да, то условие становится истинным.	<i>5 > 2 в результате будет True. True > False в результате будет True. "A" > "B" в результате будет False.</i>
<	Проверяет, меньше ли значение левого операнда чем значение правого. Если да, то условие становится истинным.	<i>3 < 5 в результате будет True. True < False в результате будет False. "A" < "B" в результате будет True.</i>
>=	Проверяет, больше или равно значение левого операнда чем значение правого. Если да, то условие становится истинным.	<i>1 >= 1 в результате будет True. 23 >= 3.2 в результате будет True. "C" >= "D" в результате будет False.</i>
<=	Проверяет, меньше или равно значение левого операнда чем значение правого. Если да, то условие становится истинным.	<i>4 <= 5 в результате будет True. 0 <= 0.0 в результате будет True. -0.001 <= -36 в результате будет False.</i>

Операторы сравнения

=	Присваивает значение правого операнда левому.	<i>b = 23 присвоит переменной b значение 23</i>
+=	Прибавит значение правого операнда к левому и присвоит эту сумму левому операнду	<i>b = 5 a = 2 b += a равносильно: b = b + a. b=7</i>
-=	Отнимает значение правого операнда от левого и присваивает результат левому операнду.	<i>b = 5 a = 2 b -= a равносильно: b = b - a. b=3</i>
*=	Умножает правый операнд с левым и присваивает результат левому операнду.	<i>b = 5 a = 2 b *= a равносильно: b = b * a. b=10</i>
/=	Делит левый операнд на правый и присваивает результат левому операнду.	<i>b = 10 a = 2 b /= a равносильно: b = b / a. b=5</i>
%=	Делит по модулю операнды и присваивает результат левому.	<i>b = 5 a = 2 b %= a равносильно: b = b % a. b=1</i>
**=	Возводит в левый операнд в степень правого и присваивает результат левому операнду.	<i>b = 3 a = 2 b **= a равносильно: b = b ** a. b=9</i>
//=	Производит целочисленное деление левого операнда на правый и присваивает результат левому операнду.	<i>b = 11 a = 2 b // a равносильно: b = b // a. b=5</i>

Арифметические операторы

Оператор	Значение	Примеры
and	Логический оператор "И". Условие будет истинным, если оба операнда истина.	<i>True and True</i> равно <i>True</i> . <i>True and False</i> равно <i>False</i> . <i>False and True</i> равно <i>False</i> . <i>False and False</i> равно <i>False</i> .
or	Логический оператор "ИЛИ". Если хотя бы один из операндов истинный, то и все выражение будет истинным.	<i>True or True</i> равно <i>True</i> . <i>True or False</i> равно <i>True</i> . <i>False or True</i> равно <i>True</i> . <i>False or False</i> равно <i>False</i> .
not	Логический оператор "НЕ". Изменяет логическое значение операнда на противоположное.	<i>not True</i> равно <i>False</i> . <i>not False</i> равно <i>True</i> .

Операторы членства

Операторы членства проверяют наличие элемента в составных типах данных, таких как строки, списки, кортежи или словари:

Оператор	Значение	Примеры
in	Возвращает истину, если элемент присутствует в последовательности, иначе возвращает ложь.	<i>"cad" in "cadillac"</i> вернет <i>True</i> . <i>1 in [2,3,1,6]</i> вернет <i>True</i> . <i>"hi" in {"hi":2,"bye":1}</i> вернет <i>True</i> . <i>2 in {"hi":2,"bye":1}</i> вернет <i>False</i> (в словарях проверяется наличие в ключах, а не в значениях).
not in	Возвращает истину, если элемента нет в последовательности.	Результаты противоположны результатам оператора <i>in</i> .

Операторы тождественности

Операторы тождественности сравнивают размещение двух объектов в памяти компьютера.

Оператор	Значение	Примеры
is	Возвращает истину, если оба операнда указывают на один объект.	<i>x is y</i> вернет истину, если <i>id(x)</i> будет равно <i>id(y)</i> .
is not	Возвращает ложь, если оба операнда указывают на один объект.	<i>x is not y</i> , вернет истину, если <i>id(x)</i> не равно <i>id(y)</i> .

Арифметические операторы

Оператор	Значение
**	Возведение в степень
$\sim + -$	Комплементарный оператор
$* / \% //$	Умножение, деление, деление по модулю, целочисленное деление
$+ -$	Сложение и вычитание
$>> <<$	Побитовый сдвиг вправо и побитовый сдвиг влево
$\&$	Бинарный "И"
$\wedge $	Бинарный "Исключительное ИЛИ" и бинарный "ИЛИ"
$<= < > >=$	Операторы сравнения
$<> == !=$	Операторы равенства
$= %= /= //=-+=*=**=$	Операторы присваивания
is is not	Тождественные операторы
in not in	Операторы членства
not or and	Логические операторы

Приложение №4

Примеры форматов растровых рисунков

ФОРМАТ	ДОСТОИНСТВА	НЕДОСТАТКИ
BMP (.bmp) стандартный формат, который	- поддерживает кодирование с палитрой и в режиме истинного цвета;	- занимает много места; - применение файлов BMP ограничено платформами Windows и OS/2., что ограничивает применение формата в сети.
JPEG (.jpg или .jpeg) – формат, разработанный специально для кодирования фотографий	- лучше других сжимает фотографии и картины, содержащих реалистичные сцены с плавными переходами яркости и цвета;	- занимает много места; - применение файлов BMP ограничено платформами Windows и OS/2., что ограничивает применение формата в сети.
GIF (.gif) – формат, поддерживающий только кодирование с палитрой (от 2 до 256 цветов); в отличие от предыдущих форматов.	- части рисунка могут быть прозрачными, то есть на веб-странице через них будет «просвечивать» фон; - поддерживает анимацию.	- малое количество цветов; - поддерживает анимационные изображения, которые представляют собой последовательность из нескольких статичных кадров, а также информацию о том, сколько времени каждый кадр должен быть показан на экране
PNG (.png) – формат, поддерживающий как режим истинного цвета, так и кодирование с палитрой; части изображения могут быть прозрачными и даже полу-прозрачными	- сжатие производится без потерь; - восстановление и пересохранение изображения проходят без потерь в качестве.	- несмотря на то что данный формат был изначально спроектирован для замены устаревшего формата, GIF, PNG может хранить только одно изображение в одном файле.

Глоссарий

Аутентификация – это процесс проверки подлинности чего-либо. Примером **аутентификации** может быть сравнение пароля, введенного пользователем, с паролем, который сохранен в базе данных сервера.

«Бот» (сокращение от «робот») – программа, имитирующая деятельность человека. Чат-бот имитирует собеседника в чате.

Видеопамять – это часть оперативной памяти, в которой формируется графическое изображение.

Гиподинамия – низкая двигательная активность организма, снижение общей физической активности.

Глубина цвета – это количество бит, используемых для кодирования цвета пикселя.

Дискретизация – это преобразование графической информации из аналоговой формы в дискретную, то есть разбиение непрерывного графического изображения на отдельные элементы.

Естественный язык – используется для общения между людьми (русский, кыргызский, английский языки).

Информационная грамотность – это способность человека осознавать необходимость в информации, умение ее искать, отбирать, оценивать и использовать.

Интерпретатор – это программа, которая читает вашу программу и выполняет содержащиеся в ней инструкции.

Контент (англ. содержание) – это информация и опыт, предназначенные для конечного пользователя. Контент сайта – это все содержимое сайта, включая текст, картинки, видео и аудио.

Код – это система условных знаков и правил для представления информации.

Кодирование – это представление информации с помощью заданного кода.

Логические выражения – это логические высказывания, соединенные логическими операциями.

Лэндинг (англ. landing page) – веб-страница, главной целью которой является совершение посетителем сайта конкретного (целевого) действия: например, проголосовать за что-то, ознакомиться с каким-либо товаром и купить его, заказать книгу.

Лонгрид (англ. longread) – длинный текст (статья, рассказ), разбитый на блоки с помощью различных мультимедийных элементов: фотографий, картинок, видео, диаграмм и т.д.

Переменная – это величина, которая имеет имя, тип и значение. Значение переменной может изменяться во время выполнения программы.

Простое логическое высказывание – это высказывание, которое нельзя уменьшить или разделить без потери смысла.

Протокол – это правила, в соответствии с которыми происходит передача информации через сеть.

Разрешение – это количество пикселей, приходящихся на дюйм размера изображения.

CMS – Система управления содержимым (контентом) (англ. Content management system, CMS) – это ПО, используемое для управления содержимым сайта.

Сложные высказывания состоят из простых и соединены логическими связками, которые соответствуют логическим операциям.

Формализация – это переход от конкретного содержания (высказывания) к формальной записи с помощью символов.

Формальный язык – искусственный язык, характеризующийся точными правилами построения выражений: ноты, грамота, азбука Морзе, символы химических элементов, языки программирования.

Фейк (от англ. *fake* – «поддельный, фальшивый») в случае с подачей информации имеет смысл – «ложная информация».

<https://new.bizdin.kg/>