МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Министерство образования Ярославской области

Администрация Некрасовского муниципального района

МБОУ Некрасовская СОШ

Подписано цифровой подписью: Петров Александр Владимирович

РАССМОТРЕНО	УТВЕРЖДЕНО
на заседании МО	Директор школы
_____	_____
	Петров А.В.
Протокол № 1	Приказ № 17
от «28» августа 2023 г.	от «01» сентября 2023 г.

Рабочая программа внеурочной деятельности

«Основы программирования на Python»

Направленность: *техническая*

для учащихся 10 классов

Срок реализации: 68 часов

Автор-составитель:

Краснова Марина Николаевна,

учитель информатики

рп. Некрасовское 2023

Информационная карта

рабочей программы внеурочной деятельности

Рабочая программа внеурочной деятельности: «Основы программирования на Python».

Учитель информатики: Краснова Марина Николаевна

Тип программы: модифицированная.

Направленность программы: техническая.

Уровень усвоения программы: стартовый.

Продолжительность усвоения программы: 68 часов.

Возрастной диапазон начала освоения программы: 16 - 17 лет (10 класс).

Форма организации образовательного процесса: групповая, индивидуальная.

Год написания: 2023 год.

Год корректировки: 2023 год.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Рабочая программа внеурочной деятельности (далее – программа) «Основы программирования на Python» является модифицированной, имеет техническую направленность и развивающий характер.

Программа «Основы программирования на Python» направлена на удовлетворение потребностей учащихся в интеллектуальном развитии, формирование и развитие логического мышления, аналитических и творческих способностей учащихся. Программа учитывает индивидуальные особенности детей, обеспечивает поддержку каждого ребенка, его интеллектуальное и техническое развитие. Программа предполагает обучение детей программированию посредством языка Python 3 (среда программирования IDLE).

Python (произносится как «Питòн» или «Пàйтон») - современный развивающийся язык программирования. Он используется во множестве реальных проектов, поэтому его изучение не пройдёт напрасно. В настоящее время язык Python поддерживается международным сообществом разработчиков.

Python – высокоуровневый язык программирования, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис языка прост и минималистичен, что хорошо подойдет для начинающих программистов. В то же время библиотека языка весьма широка, а сам Python поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное и т.д.

Интегрируемая среда программирования IDLE (Python 3) и подобные средства для работы с языком Python относятся к свободно распространяемым программным обеспечениям. Среда IDLE является кроссплатформенным (много платформенным) продуктом, что позволяет установить его на персональные компьютеры с разными операционными системами: Windows, Linux и Mac.

Python – это текстовый интерпретируемый язык программирования, что очень удобно при обучении. Он универсален, пригоден для создания самых разных программ, от текстовых процессоров до веб-браузеров. Причины выбора этого язык программирования для изучения детьми:

– Python – простой и удобный язык. По сравнению со многими другими языками читать и составлять программы на Python совсем не сложно;

– в Python есть библиотеки готовых процедур для использования в своих программах, что позволяет создавать сложные программы быстро;

– Python используется серьёзными организациями в важных проектах, например, его используют в Google, Amazon, Yandex, NASA, Intel, Hewlett-Packard и

другие. Крупнейшие интернет ресурсы - Google и YouTube разработаны с помощью языка программирования Python;

– язык Python имеет обширную область применения, например, на нём создаются расширения к графическому редактору GIMP, на Python можно программировать в офисном пакете OpenOffice.org, пишутся сценарии для пакета 3D-моделирования Blender, создаются системы управления различными контентами (Plone и MoinMoin Wiki), а также компьютерные игры;

– Python – серьёзный язык программирования. В то же время учащиеся в полной мере могут раскрыть свои творческие способности, так как с его помощью можно легко создавать игры и другие приложения.

Обучение программированию на языке Python позволяет формировать у учащихся тягу к знаниям и творчеству, дает возможность реализовать свои идеи и быть успешным.

Рабочая программа внеурочной деятельности составлена в соответствии с Федеральным Законом от 01.09.2013 № 273-ФЗ «Об образовании в Российской Федерации» (с изменениями и дополнениями).

Программа разработана в соответствии с современными методическими рекомендациями по разработке и оформлению общеобразовательных общеразвивающих программ, с учетом современных требований и норм.

Актуальность данной программы состоит в том, что активизация познавательного процесса позволяет учащимся более полно выражать свой творческий потенциал и реализовывать собственные идеи в изучаемой области знаний, создаёт предпосылки по применению информационных компетенций в других учебных курсах, а также способствует возникновению мотивации, направленной на освоение профессий, связанных с разработкой программного обеспечения. Преимуществом среды программирования Python, среди подобных, является наличие версий программного обеспечения для различных операционных систем, к тому же программа является свободно распространяемой, что очень важно для учреждений образования детей.

Языки программирования можно сравнить с иностранными языками, овладеть ими может каждый. Учиться программировать очень интересно. Результат программирования очень часто виден сразу. Кроме того, создание компьютерных игр и обучающих программ способствует развитию логики и креативного мышления. Ещё одной значимой стороной обучения программированию является спрос на рынке труда на специалистов данного направления деятельности.

В настоящее время существует необходимость создавать условия для развития способностей и самореализации учащихся через создание эффективной системы

выявления, поддержки и обучения одаренных детей, в том числе в детских объединениях технической направленности.

Концепция федеральной целевой программы развития образования на 2016 - 2020 годы в рамках мероприятий по распространению на всей территории Российской Федерации современных моделей успешной социализации детей предполагает создание интегрированных моделей общего и дополнительного образования, в частности моделей развития техносферы в деятельности образования детей исследовательской, инженерной, технической, конструкторской направленности.

На анализе социальных проблем и социальном заказе в настоящий момент в России развиваются нано технологии, электроника, механика и программирование. В доме детского творчества созданы хорошие условия для развития робототехники и компьютерных технологий.

С развитием современных информационных технологий сегодня компьютеры и компьютерные системы – неотъемлемая часть жизни нашего общества. Научившись программировать, мы можем быть не только пользователями информационных технологий, но и активными их создателями.

При создании проектов в среде Python, учащиеся осваивают множество навыков XXI века, которые будут необходимы для успеха:

- общение с единомышленниками;
- эффективное взаимодействие;
- логическое мышление;
- системный анализ;
- беглое использование технологий;
- проектирование;
- умение обучатся и самообразовываться;
- самостоятельно принимать решения.

Обучение по рабочей программе внеурочной деятельности «Основы программирования на Python» – это один из интересных способов изучения компьютерных технологий и основ программирования.

Программа ориентирована на детей среднего школьного возраста 16 - 17 лет (учащихся 10 классов).

При разработке программы учитывались возрастные особенности учащихся.

Характерная особенность детей подросткового возраста 16 - 17 лет – целеустремленность, настойчивость и импульсивность.

Подростковый возраст – это время, когда формируется осознание себя в социуме, познание норм поведения и общения. Подростка особенно интересуют социальные проблемы, ценности, закладывается жизненная позиция. Появляется стремление к самореализации своих способностей. В подростковом возрасте ребенок в состоянии дифференцировать то, что действительно ему интересно, чем бы он хотел заниматься в будущем. Подросток достигает успехов в конкретной сфере деятельности, определяющей его дальнейшую жизнь. В этот период укрепляются качества, которые являются фундаментом для его мировоззрения.

Особенности теоретического мышления позволяют подросткам анализировать абстрактные идеи, искать ошибки и логические противоречия в суждениях. Подросток умеет оперировать гипотезами, решая интеллектуальные задачи. Он способен на системный поиск решений. Сталкиваясь с новой задачей, он старается отыскать разные возможные подходы к её решению, проверяя логическую эффективность каждого из них. Находит способы применения абстрактных правил для решения целого класса задач.

Организация учебной деятельности подростков – важнейшая и сложнейшая задача. Ученик среднего школьного возраста вполне способен понять аргументацию педагога, родителя, согласиться с разумными доводами. Однако в виду особенностей мышления, характерных для данного возраста, подростка уже не удовлетворит процесс сообщения сведений в готовом, законченном виде. Ему захочется проверить их достоверность, убедиться в правильности суждений. Споры с учителями, родителями, приятелями – характерная черта данного возраста. Их важная роль заключается в том, что они позволяют обменяться мнениями по теме, проверить истинность своих воззрений и общепринятых взглядов, проявить себя. В частности, в обучении большой эффект дает внедрение проблемных задач.

Занятия по программированию на Python развивают логику, повышают системность мышления, а также развивают творческие способности. Все это так же влияет на степень осознанности в принимаемых решениях. Даже, если в будущем подросток не станет программистом, то знания, как создаются компьютерные программы, обязательно пригодятся ему в другой деятельности, какую бы профессию он не выбрал в будущем.

Педагогическая целесообразность программы заключается в привлечении учащихся к занятиям техническим творчеством, что способствует развитию логического мышления, воображения и навыков решения задач программирования. Реализация программы позволяет повысить эффективность познавательного процесса учащихся. Программа является целостной и непрерывной в течение всего процесса обучения, и позволяет учащемуся шаг за шагом раскрывать в себе творческие возможности.

Благодаря простоте языка Python дети легко ориентируются в коде программы. Они осваивают все основные алгоритмические конструкции, такие как линейные программы, циклы и ветвления. Обучение проходит в интересной форме, учащиеся могут создавать свои программы, проекты и простые игры, делиться ими с другими.

Занятия программированием решают проблему занятости детей, развивают у них такие черты характера, как: терпение, аккуратность, силу воли, упорство в достижении поставленной цели и трудолюбие. Программирование мотивирует к занятиям в различных научных областях (информатики, алгебры, геометрии и др.) и способствует ранней профориентации подростков.

Новизна программы «Основы программирования на Python» заключается в новом решении проблем дополнительного образования и основана на комплексном подходе к подготовке ребенка к получению дальнейшего образования, развитию технических и интеллектуальных способностей через использование проектных и исследовательских технологий, подготовке личности «новой формации», готового к освоению информационных технологий и языков программирования.

Содержание программы позволяет учащимся погрузиться в мир алгоритмизации и программирования на одном из популярных языков Python в доступной и увлекательной форме. Изучая программирование на Python, учащиеся смогут проявлять свои творческие способности, создавая различные программы и простые компьютерные игры.

Для развития творческих способностей учащиеся на занятиях привлекаются к проектной деятельности. В ходе работы над творческими проектами учащимся необходимо будет учиться решать определенные задачи для достижения цели. Самостоятельное выполнение практических заданий при решении определенной проблемы, дают возможность учащемуся самостоятельно выбирать пути ее решения. В процессе создания проектов учащийся не просто освоит азы программирования, но и познакомится с полным циклом разработки программы, начиная с этапа описания идеи и заканчивая тестированием и отладкой.

Программа «Основы программирования на Python» составлена с учетом тенденций развития современных информационных технологий, что позволяет сохранять актуальность реализации данной программы.

Отличительная особенность программы «Основы программирования на Python» от других рабочих программ внеурочной деятельности заключается в том, что изучение основ программирования с помощью языка Python позволяет детям создавать собственные компьютерные программы и простые игры. К тому же, данная программа направлена на

развитие у обучающихся логического и пространственного мышления, творческих способностей, и приобщает к техническому творчеству.

На занятиях учащиеся познакомятся с теоретическими аспектами и синтаксисом языка, а также обучатся практическим навыкам программирования на Python. Занятия начинаются с практического знакомства со средой программирования IDLE (Python 3), далее идет непосредственное изучение синтаксических конструкций языка и отработка навыков применения элементов программирования при решении задач и создании простых проектов и игр. Каждая новая тема завершается практическими задачами, способствующими овладению методики программирования и изучению языка Python. Обучение детей основам программирования построено по принципу от простого к сложному, а также на увеличении доли практических занятий. На практических занятиях учащиеся работают в среде программирования IDLE (Python 3). В ходе обучения предполагается не только индивидуальная, но и работа в малых группах над практическими заданиями.

Рабочая программа внеурочной деятельности «Основы программирования на Python» может помочь подростку определиться с выбором будущей профессии.

Цель программы: формирование логического мышления, творческого и познавательного потенциала подростков, повышение мотивации к изучению программирования через создание программ и проектов на языке Python.

Достижение поставленной цели возможно при решении следующих задач:

***обучающих*:**

- познакомить с основами алгоритмизации и программирования;
- научить основам программирования на языке Python;
- обучить основам работы по созданию простых компьютерных программ, творческих проектов и игр на языке программирования Python.

развивающих:

- способствовать формированию умений и навыков планирования и разработки компьютерных программ, творческих проектов и игр, самостоятельности в решении задач;
- создать условия для развития памяти, алгоритмического, логического, системного и творческого мышления;
- прививать интерес к программированию, к научно-техническому творчеству, технике и высоким технологиям.

воспитательных:

- воспитывать информационную, техническую и исследовательскую культуру учащихся;

– воспитывать настойчивость в достижении поставленной цели и трудолюбие;

– прививать коммуникативные навыки общения друг с другом и умение заниматься в коллективе.

Направленность рабочей программы внеурочной деятельности «Основы программирования на Python»: техническая.

Уровень освоения программы: стартовый.

Объем программы: 68 часов.

Срок реализации программы – 68 часов.

Периодичность и продолжительность занятий: программа рассчитана на **1** занятие в неделю по **2** академических часа**. 1** академический час – 45 минут.

Адресат программы: программа рассчитана на обучение и удовлетворяет техническим потребностям детей подросткового возраста от 16 до 17 лет (учащихся 10 классов), выразившим желание познакомиться с основами программирования на языке Python, сориентирована как на девочек, так и на мальчиков.

Наполняемость группы – 10 человек (или по количеству персональных компьютеров).

Форма обучения: очная.

Формы организации образовательного процесса: групповые и индивидуальные.

Формы и режим занятий: основной организационной формой в ходе реализации программы является учебное занятие. Данная форма обеспечивает организационную четкость и непрерывность процесса обучения. Занятия проводятся 2 раза в неделю по 1 часу в групповой форме. Каждое занятие включает в себя организационные моменты и здоровьесберегающие технологии (короткие перерывы, упражнения для улучшения осанки и для глаз, режим проветривания помещения).

Виды учебных занятий: лекции, практические занятия и творческие проекты.

В программе используются общедоступные и универсальные формы организации материала, а также минимальная сложность содержания.

Формы подведения итогов реализации программы: нулевая, промежуточная и итоговая аттестация.

Педагогические принципы реализации программы:

– принцип наглядности (на занятиях активно используется мультимедийная доска, проектор, видео ролики и обучающие программы, поскольку через органы зрения человек получает в 5 раз больше информации, чем через слух);

– принцип доступности (при изложении нового материала учитываются возрастные особенности детей. Занятия распределены по принципу: от простого к сложному. При необходимости допускается повторение пройденного ранее материала);

– принцип сознательности и активности (для активизации самостоятельной деятельности обучающихся на занятиях используются такие формы обучения, как совместные обсуждения вопросов, мозговой штурм, и возможность свободного творчества);

– принцип научности (в основу положены объективные достижения современной науки);

– принцип вариативности (возможен выбор самостоятельных практических заданий исходя из предпочтений учащегося).

В программе используются различные виды педагогических технологий: группового обучения, проблемного обучения, а также технологии исследовательской и проектной деятельности. Виды занятий определяются содержанием программы. Основной формой обучения является самостоятельная практическая работа, которая выполняется индивидуально или малыми группами.

Методы обучения:

– словесный;

– наглядный;

– практический;

– познавательный (восприятие, осмысление и запоминание учащимися нового материала с привлечением наблюдения готовых примеров, моделирования, изучения иллюстраций, восприятия, анализа и обобщения демонстрируемых материалов);

– метод проектов (при усвоении и творческом применении навыков и умений в процессе разработки собственных проектов);

– контрольный метод (при выявлении качества усвоения знаний, навыков и умений, их коррекция в процессе выполнения практических заданий).

Оценочно-результативный раздел

Ожидаемые результаты освоения содержания рабочей программы внеурочной деятельности «Основы программирования на Python».

Предметные результаты:

1. учащиеся должны знать технику безопасности и особенности работы с персональным компьютером;
2. учащиеся должны знать основные алгоритмы, принципы работы с данными разного типа, с операторами, операциями, методами и функциями, используемыми в программировании на языке Python;
3. уметь создавать простые программы, проекты и компьютерные игры на языке Python в среде программирования IDLE;
4. знать особенности проектной деятельности.

Метапредметные результаты:

1. овладение основными умениями и навыками решения творческих и технических задач;
2. овладение умением планировать, контролировать и оценивать свою деятельность в соответствии с поставленной задачей;
3. проявление интереса к техническому творчеству.

Личностные результаты:

1. принятие норм информационной, технической и исследовательской культуры;
2. развитие мотивов в учебной деятельности и саморазвития;
3. развитие навыков сотрудничества со сверстниками и взрослыми, умение презентовать себя и выступать перед аудиторией.

В рамках реализации программы учащиеся получат стартовые знания по алгоритмизации и программированию на языке Python, будут знать принципы работы с данными разного типа, с операторами, операциями, методами и функциями, используемыми в программировании на языке Python. Учащиеся узнают алгоритм разработки творческого проекта и простой компьютерной игры.

По окончании обучения, учащиеся будут уметь составлять основные алгоритмические конструкции на языке Python для решения задач. Учащиеся будут уметь реализовывать алгоритмы на компьютере в виде программ, написанных на Python, овладеют основными навыками программирования в среде IDLE (Python 3). Будут уметь планировать проект, составлять алгоритм его разработки, уметь самостоятельно искать способы решения

задач и решать их в процессе работы над созданием творческого проекта. Учащиеся научатся тестировать и отлаживать работу программ, написанных на языке Python.

Способы определения результативности. Виды и формы контроля.

Оценочные материалы Программы дифференцированы по трем уровням сложности: «достаточный» (1 уровень), «средний» (2 уровень) и «оптимальный» (3 уровень). Каждый уровень выражается в баллах. Принята 3-х балльная оценка результатов освоения программы: «достаточный» - 1 балл, «средний» - 2 балла и «оптимальный» - 3 балла.

Степень выраженности каждого показателя:

1 уровень - достаточный (освоение программы на начальном уровне);

2 уровень - средний (полное освоение программы);

3 уровень - оптимальный (полное освоение программы, высокий образовательный результат, находит оригинальные способы выполнения поставленной творческой задачи, транслирует творческие достижения на итоговом занятии). Результат оформляется в общую на группу таблицу результативности обучения учащихся – сводную таблицу итоговой аттестации учащихся (Приложение 3).

Уровни освоения программы

Показатели		**Критерии**	**Выраженность в баллах**
Предметные	**Оптимальный**	Владеет терминологией и основными понятиями алгоритмизации и принципами программирования. Владеет необходимыми знаниями: – знает и соблюдает технику безопасности при работе с персональным компьютером; – знает назначение и возможности среды программирования IDLE (Python 3) и принципы работы с ней; – знает основы алгоритмизации и алгоритмические конструкции; – знает основы программирования на языке Python; – знает основные алгоритмы, принципы работы с данными разного типа, с операторами, операциями, методами и функциями, используемыми в программировании на языке Python; – знает особенности работы над простыми проектами на языке Python; – знает алгоритм разработки проекта. Умеет применять полученные знания в практической деятельности. Умеет самостоятельно создавать простые программы, проекты и компьютерные игры на языке Python в среде программирования IDLE; Умеет создавать объекты для проекта и программировать их на выполнение конкретных задач. Умеет самостоятельно решать технические задачи в процессе работы над проектами. Умеет разрабатывать творческие проекты на заданную или свободную тематику. Интересуется дополнительным материалом.	3

	Средний	Владеет терминологией и некоторыми понятиями алгоритмизации и принципами программирования. Владеет необходимыми знаниями: – знает и соблюдает технику безопасности при работе с персональным компьютером; – знает назначение и возможности среды программирования IDLE (Python 3) и принципы работы с ней; – знает основы алгоритмизации и алгоритмические конструкции; – знает основы программирования на языке Python; – знает основные алгоритмы, принципы работы с данными разного типа, с операторами, операциями, методами и функциями, используемыми в программировании на языке Python; – знает особенности работы над простыми проектами на языке Python, но для составления алгоритма работы над проектом прибегает к помощи педагога; Умеет применять некоторые полученные знания в практической деятельности. Умеет создавать простые программы, проекты и компьютерные игры на языке Python в среде программирования IDLE, но во время работы над проектами допускает ошибки, прибегает к помощи педагога. Умеет создавать объекты для проекта и программировать их на выполнение конкретных задач, с незначительными недочетами и ошибками. Для решения технических задач в процессе работы над проектами прибегает к помощи товарищей или педагога. Умеет при поддержке педагога разрабатывать творческие проекты на заданную или свободную тематику.	2
	Достаточный	Владеет некоторыми терминами и понятиями алгоритмизации и принципами программирования. Усвоил некоторые знания: – знает и соблюдает технику безопасности при работе с персональным компьютером; – знает назначение и некоторые возможности среды программирования IDLE (Python 3) и принципы работы с ней; – знает принципы алгоритмизации и некоторые алгоритмические конструкции; – знает принципы программирования на языке Python; – знает некоторые алгоритмы и понимает основные принципы работы с данными разного типа, с операторами, операциями, методами и функциями, используемыми в программировании на языке Python; – знает, как создаются простые проекты на языке Python, но во время выполнения практических заданий постоянно прибегает к помощи педагога; Умеет применять некоторые полученные знания в практической деятельности. Постоянно прибегает к помощи товарищей или педагога во время работы над созданием простых программ, проектов и компьютерных игр на языке Python в среде программирования IDLE, но во время работы над проектами допускает ошибки. Умеет создавать объекты для проекта и программировать их на выполнение конкретных задач, с незначительными недочетами и ошибками. Для решения технических задач в процессе работы над проектами постоянно прибегает к помощи товарищей или педагога. При работе над проектами на заданную или свободную тематику требуется контроль со стороны педагога.	1

Метапредметные	**Оптимальный**	Владеет целеполаганием, умеет поставить цель и достигает её. Владеет основными умениями и навыками решения задач при работе над проектами. Умеет планировать практическую деятельность на занятии. Умеет использовать знания, понятия, усвоенные на других занятиях и в самостоятельной работе при работе над проектами. Предлагает оригинальные технологические приёмы при работе над интерактивными проектами. Умеет контролировать и оценивать свою деятельность в соответствии с поставленной задачей. Умеет самостоятельно разрабатывать несложные проекты и реализовывать их, вносить коррективы в полученные результаты. Проявляет интерес к техническому творчеству. Умеет правильно оценивать свою деятельность на занятии. Имеет способность к саморазвитию и самообразованию.	3
	Средний	Владеет целеполаганием, умеет поставить цель и достигает её. Владеет основными умениями и навыками решения задач при работе над проектами. Умеет планировать практическую деятельность на занятии. Умеет разрабатывать несложные проекты и с помощью педагога их реализовывать, вносить коррективы в полученные результаты. Предлагает простые технологические приёмы при работе над интерактивными проектами. Умеет совместно с педагогом и другими детьми давать оценку деятельности на занятии. Умеет разрабатывать простые проекты и реализовывать их, прибегает к помощи педагога. Проявляет интерес к техническому творчеству.	2
	Достаточный	С помощью педагога может поставить цель и достигает её. С помощью педагога умеет планировать практическую деятельность на занятии. Умеет совместно с педагогом и другими детьми давать оценку деятельности на занятии.	1
Личностные	**Оптимальный**	Принимает нормы информационной, технической и исследовательской культуры. Имеет внутреннюю мотивацию в учебной деятельности и саморазвитии. Проявляет интерес к содержанию программы, исследовательской и проектной деятельности. Чувствует уверенность в своих возможностях. Обладает навыками сотрудничества со сверстниками и взрослыми. Умеет презентовать себя и выступать перед аудиторией. Умеет самостоятельно определять и объяснять свои чувства и ощущения, возникающие в результате наблюдения, рассуждения, обсуждения. Умеет соблюдать общие для всех людей правила поведения (основы общечеловеческих нравственных ценностей), отзывчиво относится к товарищам, проявляет готовность оказать им посильную помощь. Проявляет интерес к достижениям науки в области информационных технологий.	3
	Средний	Принимает нормы информационной, технической и исследовательской культуры. Имеет внутреннюю мотивацию в учебной деятельности. Проявляет интерес к содержанию программы, исследовательской и проектной деятельности. Чувствует удовлетворение от выполненной работы. Обладает навыками сотрудничества со сверстниками и взрослыми. Умеет презентовать себя и выступать перед аудиторией.	2

		Умеет с помощью педагога определять и объяснять свои чувства и ощущения, возникающие в результате наблюдения, рассуждения, обсуждения. Умеет соблюдать общие для всех людей правила поведения (основы общечеловеческих нравственных ценностей), отзывчиво относится к товарищам, проявляет готовность оказать им посильную помощь. Проявляет внимание к беседам про достижения науки в области информационных технологий.	
	Достаточный	Принимает нормы информационной, технической и исследовательской культуры. Имеет внутреннюю мотивацию в учебной деятельности. Положительно относится к занятиям, но не проявляет интерес к исследовательской и проектной деятельности. Чувствует удовлетворение от выполненной работы. Умеет с помощью педагога определять и объяснять свои чувства и ощущения, возникающие в результате наблюдения, рассуждения, обсуждения. Умеет соблюдать общие для всех людей правила поведения (основы общечеловеческих нравственных ценностей), отзывчиво относится к товарищам. Принимает помощь, положительно отзывается на помощь взрослых и детей. Проявляет внимание к беседам про достижения науки в области информационных технологий.	1

Для выявления итоговой оценки уровня освоения ДОП необходимо суммировать баллы, если количество набранных баллов составляет:

8-9 баллов – оптимальный уровень; 5-7 баллов – средний уровень; 3-4 балла – достаточный уровень. По окончании обучения учащиеся получат основные навыки работы создания простых программ, проектов и компьютерных игр на языке Python. Обязательным для каждого обучающегося является создание различных программных продуктов на языке Python.

В период обучения проводится фиксация достижений учащихся в виде входящей (нулевой), промежуточной и итоговой аттестации. Для отслеживания результативности образовательного процесса в качестве нулевой аттестации проводится устный опрос, во время проведения промежуточной аттестации учащимся предлагается самостоятельная практическая работа, а также проводится тестирование.

Программа предназначена для использования в работе дополнительного образования, организаций среднего полного и общего образования, обеспечена дидактическими материалами и методическими пособиями.

Программа вариативна, возможны изменения и дополнения в учебном, учебно–тематическом плане или календарном учебном графике.

Кабинет информатики, в котором проводятся занятия по данной программе, соответствует требованиям материального и программного обеспечения. Кабинет информатики оборудован согласно правилам пожарной безопасности.

УЧЕБНЫЙ ПЛАН

№	Название раздела	Количество часов			Формы контроля
		Всего	Теория	Практика	
1	Введение в программу. Техника безопасности. Основы алгоритмизации и программирования. Знакомство с языком программирования Python.	6	2	4	Нулевой срез. Устный опрос. Беседа. Педагогическое наблюдение за деятельностью детей.
2	Типы данных. Переменные. Числа. Операторы. Процедуры.	8	4	4	Беседа. Самостоятельная практическая работа.
3	События. Методы. Функции. Ветвления. Условные операторы. Модуль random. Простые проекты на языке Python.	8	4	4	Беседа. Самостоятельное выполнение практического задания. Самооценка работы.
4	Циклы. Списки. Словари. Проекты с использованием циклов, списков и словарей.	10	4	6	Беседа. Самостоятельное выполнение практического задания. Промежуточный срез. Тестирование. Рефлексия.
5	Компьютерная графика в Python. Модули turtle и graph. Анимация движения.	10	4	6	Беседа. Самостоятельное выполнение практического задания. Самооценка работы.
6	Модуль tkinter. Разработка интерфейса программы на языке Python.	12	5	7	Беседа. Самостоятельное выполнение практического задания. Самооценка работы.
7	Разработка простых игр на Python. Итоговый проект – игра.	14	6	8	Беседа. Самостоятельная практическая работа. Итоговая диагностика. Демонстрация разработанной игры. Рефлексия.
	Итого:	**68**	**29**	**39**	

СОДЕРЖАТЕЛЬНЫЙ РАЗДЕЛ ПРОГРАММЫ

1. Введение в программу. Техника безопасности. Основы алгоритмизации и программирования. Знакомство с языком программирования Python (6 часов):

Теория: Введение в программу. Инструктаж по технике безопасности в компьютерном классе и при работе с персональным компьютером. Мир современных информационных технологий. Перспективы развития информационных технологий. Основы алгоритмизации и программирования. Основные понятия. Обзор существующих языков программирования. Виды алгоритмов: линейные, разветвляющиеся и циклические. Алгоритмические конструкции (блок-схемы). Общие сведения о языке Python. Знакомство со средой программирования IDLE (Python 3). Инструктирование по установке программы IDLE на компьютер. Рассказ и демонстрация функциональности основных элементов интерфейса программы IDLE. Режимы работы с Python. Структура программы на языке Python. Комментарии. Первые диалоговые программы на языке Python.

Практика: Создание личной папки на компьютере для будущих проектов. Построение блок-схем по алгоритмам. Составление и запись алгоритма для исполнителя. Установка программы IDLE (Python 3) на компьютер. Изучение основных элементов интерфейса среды программирования IDLE. Работа со справочной системой IDLE. Создание простейших диалоговых программ на языке Python. Сохранение программ на компьютере.

Контроль: Нулевой срез. Устный опрос. Беседа. Педагогическое наблюдение за деятельностью детей.

2. Типы данных. Переменные. Числа. Операторы. Процедуры (8 часов):

Теория: Правила программирования на языке Python. Понятия тестирования и отладки программы. Показ и разбор программ и проектов, созданных на языке Python. Типы данных. Операции с данными. Изменение типа данных. Скрипты для ввода и вывода данных. Переменные и выражения. Константы. Имена переменных и ключевые слова. Числа. Элементарные действия с числами. Задачи на элементарные действия с числами. Арифметические выражения. Деление нацело. Вывод остатка от деления. Вычисления и переменные. Операторы. Оператор присваивания. Выражения и операции. Порядок выполнения операций. Ввод данных с клавиатуры. Консольный ввод. Вывод данных на экран. Оператор присваивания. Процедуры. Создание простых программ на Python по примерам педагога. Программа «Создание персонажа». Программа «Сложение чисел и строк».

Практика: Создание простых программ с использованием простейших команд. Практические работы с переменными, выражениями, с числами. Решение задач на элементарные действия с числами. Практическая работа с использованием оператора присваивания. Практическая работа по вводу данных с клавиатуры и выводу данных на экран. Практическая работа с арифметическими выражениями (сложение, вычитание, умножение, деление, вывод остатка от деления). Программирование, тестирование и отладка программ. Работа над созданием простых программ на Python по примерам педагога. Выполнение самостоятельной практической работы над созданием программ на языке Python. Создание программы «Создание персонажа». Создание программы «Сложение чисел и строк». Программирование, тестирование и отладка программ. Сохранение программ на компьютере.

Контроль: Беседа. Самостоятельная практическая работа.

3. События. Методы. Функции. Ветвления. Условные операторы. Модуль random. Простые проекты на языке Python (8 часов):

Теория: События. Обработка событий. Управление событиями. Команды и их категории. Методы. Функции. Ветвления. Команды ветвления. Условные операторы. Операторы if, elif и else. Логический тип данных. Логические переменные. Операции И, ИЛИ, НЕ. Вложенные условные операторы. Неполная форма условного оператора. Сложные условные выражения. Порядок выполнения операций. Решение практических задач (составление программ) с использованием разных условных операторов, методов, функций и операций. Создание программы «Принятие решения». Простые проекты на языке Python. Проект «Замок дракона». Создание проекта «Экспертная система». Понятие «случайное число». Случайные и псевдослучайные числа. Использование модуля random. Проекты с использованием модуля random для выбора случайных чисел. Проект «Угадай число». Проект «Генератор случайных чисел».

Практика: Составление программ с использованием разных условных операторов, методов, функций и операций. Работа над созданием программы «Принятие решения». Работа над созданием простых проектов на языке Python. Работа над созданием проекта «Замок дракона». Работа над созданием проекта «Экспертная система». Выполнение самостоятельного практического задания по созданию простых проектов с использованием модуля random для выбора случайных чисел. Работа над созданием проекта «Угадай число». Работа над созданием проекта «Генератор случайных чисел». Программирование, тестирование и отладка программ. Сохранение программ на компьютере.

Контроль: Беседа. Самостоятельное выполнение практического задания. Самооценка работы.

4. Циклы. Списки. Словари. Проекты с использованием циклов, списков и словарей (10 часов):

Теория: Понятие цикла. Тело цикла. Оператор цикла с условием. Оператор цикла while. Бесконечные циклы. Оператор цикла с параметром for. Операторы управления циклом. Вложенные циклы. Цикл с предусловием. Циклы с переменной. Преобразование одной инструкции цикла в другую. Программа «Сделать N раз». От цикла while к циклу for. Оператор прерывания цикла break. Примеры использования циклов. Задачи с циклами. Проект с использованием цикла for «Таблица умножения». Списки. Тип списка (list). Элементы и индексы списков. Операторы для списков. Создание списков и их использование. Обход списков. Замена, удаление и добавление элементов списка. Объединение списков. Проект с использованием списков «Последовательности». Словари. Ячейка словаря с меткой key. Создание словаря и их использование. Проекты с использованием словарей. Проект (программа-переводчик) «Переводчик с инопланетного». Создание собственной шифровальной программы со своим шифром. Проект «Шифр Цезаря». Промежуточная аттестация: тестирование по основным понятиям языка Python и первичным навыкам программирования в среде IDLE (Python 3) (тест «Основы языка Python»).

Практика: Решение задач с циклами. Составление программ с использованием циклов, списков и словарей. Работа над созданием проекта с использованием цикла for «Таблица умножения». Работа над созданием проекта с использованием списков «Последовательности». Программирование. Тестирование и отладка программ. Выполнение самостоятельных практических заданий. Работа над созданием проектов с использованием словарей. Работа над проектом «Переводчик с инопланетного». Придумывание своего шифра и работа над проектом «Шифр Цезаря». Программирование. Тестирование и отладка программ. Сохранение программ на компьютере. Прохождение теста «Основы языка Python». Рефлексия.

Контроль: Беседа. Самостоятельное выполнение практического задания. Промежуточный срез. Тестирование. Рефлексия.

5. Компьютерная графика в Python. Модули turtle и graph. Анимация движения (10 часов):

Теория: Компьютерная графика в Python. Понятие компьютерной графики. Модули turtle и graph. Модуль turtle - «рисующая черепашка». Рисование с помощью

черепашки. Стили и цвета черепашки. Перемещения черепашки по пикселям. Управление пикселями. Работа с цветами. Создание простых рисунков. Рисование различных фигур с помощью модуля turtle. Создание рисунка «Дом». Рисование разных снежинок. Создание сложных рисунков с помощью черепашки. Рисование цветных спиралей и орнаментов с помощью модуля turtle. Модуль graph. Система координат. Координатная плоскость. Точка отсчёта, оси координат, единица измерения расстояния, определение координат. Команда переместиться в точку с заданными координатами. Скорость рисования объектов. Использование функций и циклов в компьютерной графике для создания программ для рисования фигур, узоров и т.д. Рефакторинг. Рисование с помощью модуля graph. Рисуем линии, прямоугольники, окружности. Изменение координат. Создание рисунков в Python. Использование процедур в создании рисунков. Штриховка разными вариантами. Создание фона для рисунка. Создание программы для рисования и штриховки разных фигур с помощью модуля graph. Анимация движения. Принципы создания простой анимации с использованием модуля tkinter. Проект «Прыгающий мячик».

Практика: Выполнение самостоятельных практических заданий по созданию простых и сложных рисунков с помощью модуля turtle - «рисующей черепашки». Написание программ для рисования различных фигур с помощью модуля turtle. Работа над созданием рисунка «Дом». Работа над созданием программ, рисующих снежинки, цветные спирали и орнаменты с помощью модуля turtle. Написание программ для рисования и штриховки различных фигур с помощью модуля graph. Программирование, тестирование и отладка программ. Работа над созданием проекта с использованием анимации движения «Прыгающий мячик». Программирование. Тестирование и отладка программы.

Контроль: Беседа. Самостоятельное выполнение практического задания. Самооценка работы.

6. Модуль tkinter. Разработка интерфейса программы на языке Python (12 часов):

Теория: Модуль tkinter, его возможности и назначение. Графический пользовательский интерфейс (GUI) программы, разрабатываемой в среде IDLE (Python 3). Окна, кнопки и другие элементы графического интерфейса программы. Размеры окон. Использование системы координат и пикселей для расположения кнопок и других элементов в окне программы. Использование функций для работы с кнопками и другими

элементами окна программы на языке Python. Проект «Кнопка сюрприз». Проект «Определение победителя». Проект «Программа для рисования».

Практика: Выполнение самостоятельных практических заданий по созданию элементов интерфейса программы. Работа над созданием Разработка графического пользовательского интерфейса (GUI) программы на языке Python. Работа над созданием проекта «Кнопка-сюрприз». Работа над созданием проекта «Определение победителя». Работа над созданием проекта «Программа для рисования». Создание холста для проекта. Использование переменных и различных функций для отслеживания положения указателя компьютерной мыши, для её перемещения и возможности рисования ею на холсте. Программирование. Тестирование и отладка программы.

Контроль: Беседа. Самостоятельное выполнение практического задания. Самооценка работы.

7. Разработка простых игр на Python. Итоговый проект – игра (14 часов):

Теория: Беседа на тему создания собственных проектов или игр на языке Python. Принципы разработки простых игр на языке Python. Этапы создания игр. Применение анимации при создании игр в среде IDLE. Создание игр по шаблонам педагога с возможностью создания собственных объектов, применения нестандартных решений и фонов. Игра «Поймай шарик». Игра «Змейка». Игра «Крестики-нолики», Игра «Камень. Ножницы. Бумага». Игра «Сапер». Особенности создания игр для двоих пользователей. Итоговый проект - игра для двух пользователей с графическим интерфейсом (игра разрабатывается в парах). Выстраивания траектории вылета мяча и взаимодействия противников. Варианты составления подпрограммы. Последовательность движений мяча.

Практика: Выполнение самостоятельных практических работ. Работа над созданием игр: «Поймай шарик», «Змейка», «Крестики-нолики», «Камень. Ножницы. Бумага» и «Сапер». Программирование. Тестирование и отладка программ. Испытание игр в действии. Разработка итогового проекта игры для двоих игроков на языке Python. Составление алгоритма работы программы. Создание интерфейса игры. Создание объектов и действий для объектов игры. Составление подпрограммы. Программирование. Тестирование и отладка программы. Демонстрация готовой игры на итоговом занятии (игра-соревнование на компьютере в парах).

Контроль: Беседа. Самостоятельная практическая работа. Итоговая диагностика. Демонстрация разработанной игры. Рефлексия.

Организационно-методический раздел

Условия реализации программы

Программа соответствует требованиям нормативно-правовых документов.

Нормативно-правовое обеспечение Программы «Основы программирования на Python»:

– Федеральный Закон от 01.09.2013 № 273-ФЗ «Об образовании в Российской Федерации» (с изменениями и дополнениями);

Программа составлена в соответствии с требованиями и нормами:

– Письмо Министерства образования и науки Российской федерации от 11.12.2006 г. N 06-1844 «О примерных требованиях к программам дополнительного образования детей»;

– Приказ Министерства просвещения Российской Федерации от 09.11.2018 г. № 196 г. Москва «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»;

– Методические рекомендации для субъектов Российской Федерации по вопросам реализации основных и дополнительных общеобразовательных программ в сетевой форме, направленные письмом Министерства Просвещения РФ от 26.06.2019 №03-1235;

– СанПин 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»;

Материально-техническое обеспечение:

Кабинет на 10 рабочих мест, оборудованный стационарными персональными компьютерами (монитор, системный блок, клавиатура, мышь) или ноутбуками с программным обеспечением, столами, стульями, общим освещением. Интерактивная доска или экран. Мультимедийный проектор. Цветной струйный принтер. Сканер. Колонки. Программное обеспечение для ПК: операционная система Windows, Linux и Mac (по возможностям), Программное обеспечение для ПК среда программирования IDLE (Python 3).

Инструменты: канцелярские принадлежности и расходные материалы для принтера: цветные чернила.

Материалы: бумага, тетради.

Наглядные пособия: пошаговые инструкции для практических заданий, компакт-диски с обучающими уроками по основам программирования на языке Python, мультимедийные презентации, видеоуроки «Работа на языке Python» (видеоролики, flash-ролики).

Кадровое обеспечение – учитель информатики или педагог дополнительного образования. Специалист в области дополнительного образования детей должен ориентироваться в вопросах общей педагогики, в вопросах технического творчества. Квалификация педагогических кадров должна соответствовать утвержденному профессиональному стандарту «Педагог дополнительного образования детей и взрослых». У педагога ДО приветствуется дополнительное образование или курсы повышения квалификации в области технического творчества.

Методическое обеспечение

Для упрощения восприятия учащимися теоретического и наглядного материала разработан цикл презентаций по темам Программы «Основы программирования на Python», цикл бесед, лекций, практических заданий и т.п.

Презентации (Приложение № 4)

Презентация «ТБ в компьютерном классе и ТБ при работе с ПК».

Презентация «Алгоритмы. Алгоритмические конструкции».

Презентация «Знакомство с языком программирования Python».

Презентация «Простейшие программы на языке Python».

Презентация «Типы данных».

Презентация «Числа. Арифметические выражения».

Презентация «Символьные строки».

Презентация «Циклы. Ветвления. Условные операторы».

Презентация «Процедуры в Python».

Презентация «Методы и функции в Python».

Презентация «Алгоритм разработки проекта».

Беседы

Беседа «Правила техники безопасности при работе с персональным компьютером».

Беседа «Современные технологии и их перспективы».

Беседа «Профессия программист».

Беседа «Как создаются проекты».

Картотека с памятками по программированию, конспекты занятий, распечатки заданий для практикумов, задачи на языке Python, сборник практических работ на языке Python. Карточки с индивидуальными заданиями. Примеры игр, созданных на языке Python (Приложение № 5).

Тест по проверке знаний для промежуточной аттестации: «Основы языка Python». (Приложение 2).

Упражнения, уменьшающие усталость при работе за компьютером:

Упражнения для улучшения осанки: «Глядя в небо», «Египтянин», «Абра-кадабра».

Комплекс упражнений для глаз (Приложение 6)**.**

В рамках реализации рабочей программы внеурочной деятельности «Программирование на Python» используются типовые занятия.

Структура различных типов занятий

Тип занятия	**Основные элементы структуры занятия**
Вводное занятие	• Организационная часть • Презентация по теме • Знакомство с инструментами, необходимыми для работы. • Инструктаж по технике безопасности • Окончание занятия.
Комбинированное занятие	• Организационная часть • Проверка знаний ранее изученного материала • Изложение нового материала. • Первичное закрепление новых знаний, применение их на практике. • Окончание занятия.
Занятие сообщения и усвоения новых знаний	• Организационная часть • Теоретическая часть. Изложение нового материала. • Практическая часть. Закрепление нового. • Окончание занятия.
Занятие повторения и обобщения полученных знаний	• Организационная часть • Постановка проблем и выдача заданий. Выполнение учащимися заданий и решения задач. • Анализ ответов и оценка результатов работы, исправление ошибок. • Подведение итогов. • Окончание занятия.
Занятие закрепления знаний, выработки умений и навыков	• Организационная часть • Определение и разъяснение цели занятия. Воспроизведение учащимися знаний, связанных с содержанием предстоящей работы. • Сообщение и содержание задания, инструктаж его выполнения. • Самостоятельная работа учащихся под руководством педагога. • Обобщение и оценка выполненной работы. • Окончание занятия.

Занятие применения знаний, умений и навыков Практикум.	• Организационная часть • Определение и разъяснение целей занятия. • Установление связи с ранее изученным материалом. • Инструктаж по выполнению работы. • Самостоятельная работа учащихся. • Итоговый этап. Оценка и самооценка результатов работы. • Окончание занятия.
Занятие – тематический контроль	• Организационная часть • Тематический контроль: метод наблюдения, тестирование, выполнение практической работы, работа по схемам. • Оценка и самооценка, презентация своей работы. • Окончание занятия.

СПИСОК ЛИТЕРАТУРЫ

1. Бизли Д. Python. Подробный справочник. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с.
2. Брайсон Пэйн. Python для детей и родителей. Играй и программируй. СПб.: БХВ - Петербург, 2018. - 442с.
3. Коэльё Л. П., Ричерт В. Построение систем машинного обучения на языке Python.- Перевод с английского. — М.: ДМК Пресс, 2015.- 382с.
4. Лутц М. Изучаем Python, 4 издание / М. Лутц - СПб.: Символ-Плюс, 2011. - 1280 с.
5. Доусен М. Программируем на Python / М. Доусен - СПб.: Питер, 2016. - 416с.
6. Любанович Б. Простой Python. Современный стиль программирования / Б. Любанович. - СПб.: Питер, 2016. - 480с.
7. Маккинли У. Python и анализ данных.- Перевод с английского.- М.: ДМК Пресс, 2015.— 482с.
8. Прохоренок Н.А., Дронов В.А. Python 3 и PyQt 5. Разработка приложений / Н.А. Прохоренок, В.А. Дронов - СПб.: «БХВ- Петербург», 2016. - 832с.
9. Саммерфилд М. Python на практике.- Перевод с английского. А.А. Слинкин – М. ДМК Пресс, 2014.- 338с.
10. Сэнд У., Сэнд К. Hello World! Занимательное программирование. — СПб.: Питер, 2016. — 400 с.
11. Портал Язык программирования Python. Россум Г., Дж. Дрейк Ф.Л., Откидач Д.С., Задка М., Левис М., Монтаро С., Реймонд Э.С., Кучлинг А.М., Лембург М.-А., Йи К.-П., Ксиллаг Д., Петрилли Х.Г., Варсав Б.А., Ахлстром Дж.К., Роскинд Дж., Шеменор Н., Мулендер С.: [Электронный ресурс]. - Режим доступа: https://goo.gl/8TzY8w (дата обращения: 21.10.2019)
12. Программирование на Python / ПИТОНТЬЮТОР: [Электронный ресурс]. - Режим доступа: http://pythontutor.ru (дата обращения: 18.10.2019).
13. Официальный сайт языка Python [Электронный ресурс]. - Режим доступа: https://www.python.org (дата обращения: 23.10.2019)
14. Портал Python 3 для начинающих. [Электронный ресурс]. - Режим доступа: https://pythonworld.ru (дата обращения: 25.10.2019)

15. Сайт Интерактивный учебник языка Питон [Электронный ресурс]. - Режим доступа: http://pythontutor.ru (дата обращения: 12.11.2019)

16. Сайт Интерактивный Python. Trinket [Электронный ресурс]. - Режим доступа: https://trinket.io/python (дата обращения: 15.11.2019)

СПИСОК ЛИТЕРАТУРЫ, РЕКОМЕНДУЕМЫЙ УЧАЩИМСЯ

1. Брайсон Пэйн. Python для детей и родителей. Играй и программируй. СПб.: БХВ - Петербург, 2018. - 442с.

2. Иллюстрированное руководство по языкам Python и Python «Программирование для детей»/К. Вордерман, Дж. Вудкок, Ш. Макаманус и др.; пер. с англ. С. Ломакин. – М.: Манн, Иванов и Фербер, 2015. – 482с.

3. Программирование для детей на языке Python. - Перевод с английского А. Банкрашков — М.: Издательство АСТ — 2017. — 95с.

4. Программирование на Python / ПИТОНТЬЮТОР: [Электронный ресурс]. - Режим доступа: http://pythontutor.ru (дата обращения: 18.10.2019).

Приложение № 1

КАЛЕНДАРНЫЙ УЧЕБНЫЙ ГРАФИК

рабочей программы внеурочной деятельности

«Основы программирования на Python»

(программа стартового уровня)

Количество учебных недель: 34

Количество занятий в неделю: 1

Количество часов в учебном занятии: 2

№	№ занятия	Тема занятия	Всего часов	теория	практика	Дата проведения
РАЗДЕЛ 1. Введение в программу. Техника безопасности. Основы алгоритмизации и программирования. Знакомство с языком программирования Python.						
1	1	Введение в программу. Техника безопасности. Современные информационные технологии. **Нулевой срез. Устный опрос.** Алгоритмы. Алгоритмические конструкции. Блок-схемы.	2	1	1	
2	2	Языки программирования. Основы программирования. Язык Python. Знакомство со средой программирования IDLE (Python 3). Режимы работы с Python.	2	1	1	
3	3	Структура программы на языке Python. Первые диалоговые программы на Python.	2		2	
		Итого по теме	**6**	**2**	**4**	
РАЗДЕЛ 2. Типы данных. Переменные. Числа. Операторы. Процедуры.						
4	1	Правила программирования на Python. Понятия тестирования и отладки программы. Типы данных. Операции с данными. Команды (скрипты) на языке Python. Решение задач.	2	1	1	
5	2	Переменные и выражения. Константы. Числа. Арифметические выражения. Вычисления и переменные. Создание простых программ на языке Python. Решение задач.	2	1	1	
6	3	Операторы. Выражения и операции. События. Создание простых программ на языке Python. Решение задач.	2	1	1	
7	4	Создание простых программ на языке Python. Программа «Создание персонажа». Создание простых программ на языке Python. Программа «Сложение чисел и строк».	2	1	1	
		Итого по теме	**8**	**4**	**4**	
РАЗДЕЛ 3. События. Методы. Функции. Ветвления. Условные операторы.						

Модуль random. Простые проекты на языке Python.						
8	1	События. Управление событиями на языке Python. Команды и их категории. Методы. Функции. Ветвления. Программа «Принятие решения».	2	1	1	
9	2	Условные операторы. Операторы if, elif и else. Логический тип данных. Логические переменные. Операции И, ИЛИ, НЕ. Простые проекты на языке Python. Проект «Замок дракона».	2	1	1	
10	3	Вложенные условные операторы. Неполная форма условного оператора. Сложные условные выражения. Проект «Экспертная система».	2	1	1	
11	4	Понятие «случайное число». Случайные и псевдослучайные числа. Использование модуля random. Проект «Угадай число». Проект с использованием модуля random для выбора случайных чисел «Генератор случайных чисел».	2	1	1	
		Итого по теме	**8**	**4**	**4**	
РАЗДЕЛ 4. Циклы. Списки. Словари. Проекты с использованием циклов, списков и словарей.						
12	1	Циклы. Оператор цикла while. Бесконечные циклы. Оператор цикла с параметром for. Вложенные циклы. Цикл с предусловием. Циклы с переменной. Решение задач с циклами. Программа «Сделать N раз».	2	1	1	
13	2	От цикла while к циклу for. Оператор прерывания цикла break. Проект с использованием цикла for «Таблица умножения».	2	1	1	
14	3	Списки. Элементы и индексы списков. Операторы для списков. Работа со списками. Проект с использованием списков «Последовательности».	2	1	1	
15	4	Словари. Ячейка словаря с меткой key. Создание словаря и их использование. Проект (программа-переводчик) «Переводчик с инопланетного».	2	1	1	
16	5	Создание собственной шифровальной программы со своим шифром. Проект «Шифр Цезаря». **Промежуточный срез. Тестирование.**	2		2	
		Итого по теме	**10**	**4**	**6**	
РАЗДЕЛ 5. Компьютерная графика в Python. Модули turtle и graph. Анимация движения.						
17	1	Компьютерная графика в Python. Модули turtle и graph. Модуль turtle - «рисующая черепашка». Стили и цвета черепашки. Создание простых рисунков с помощью черепашки. Рисование различных фигур. Рисунок «Дом».	2	1	1	

18	2	Создание сложных рисунков с помощью черепашки. Рисование разных снежинок. Рисование цветных спиралей и орнаментов с помощью модуля turtle.	2	1	1	
19	3	Модуль graph. Система координат. Перемещение в точку с заданными координатами. Скорость рисования объектов. Создание рисунков с помощью модуля graph. Использование функций, циклов и процедур в компьютерной графике. Рефакторинг.	2	1	1	
20	4	Рисование с помощью модуля graph различных фигур. Штриховка фигур разными вариантами. Создание программы для рисования и штриховки разных фигур с помощью модуля graph.	2	1	1	
21	5	Анимация движения. Принципы создания простой анимации. Проект «Прыгающий мячик».	2		2	
		Итого по теме	**10**	**4**	**6**	
РАЗДЕЛ 6. Модуль tkinter. Разработка интерфейса программы на языке Python.						
22	1	Модуль tkinter, его возможности и назначение. Графический пользовательский интерфейс (GUI) программы, разрабатываемой в среде IDLE (Python 3). Элементы графического интерфейса и принцип их расположения в окне программы.	2	1	1	
23	2	Использование функций для работы с кнопками и другими элементами окна программы на языке Python. Проект «Кнопка сюрприз».	2	1	1	
24	3	Проект «Определение победителя».	2	1	1	
25	4	Проект «Программа для рисования». Использование переменных для рисования указателем мыши. Использование различных функций для рисования указателем мыши.	2	1	1	
26	5	Создание интерфейса программы для рисования.	2	1	1	
27	6	Доработка проекта «Программа для рисования». Тестирование и отладка программы.	2		2	
		Итого по теме	**12**	**5**	**7**	
РАЗДЕЛ 7. Разработка простых игр на Python.						
28	1	Принципы разработки простых игр на языке Python. Этапы создания игр. Применение анимации при создании игр в среде IDLE. Игра «Поймай шарик».	2	1	1	
29	2	Создание игр по шаблонам педагога с возможностью создания собственных объектов, применения нестандартных решений и фонов. Игра «Змейка».	2	1	1	

30	3	Игра «Крестики-нолики».	2	1	1	
31	4	Игра «Камень. Ножницы. Бумага».	2	1	1	
32	5	Игра «Сапер».	2	1	1	
33	6	Особенности создания игр для двоих пользователей.	2	1	1	
34	7	**Итоговая диагностика. Демонстрация разработанной игры.**	2		2	
Итого по теме			**14**	**6**	**8**	
ИТОГО			**68**	**29**	**39**	

Приложение № 2

Тест: «Основы языка Python»

Пройдите тест и узнайте, насколько хорошо вы знаете основные понятия языка Python и определите свои навыки программирования в среде IDLE (Python 3).

Проверка теоретических знаний:

1. Программа Python называется … 1
2. Расширение файла Python – as. …2
3. Переменная в Python – это …3
4. Регистр букв в идентификаторах значение …4
5. Выражение в Python – это …5
6. Символ # в Python обозначает …6
7. Операция 3**4 – это 8
8. 345 - … тип данных. 9
9. Операция 46%10 – это … 10
10. Функция input() – предназначена для … 12
11. Для вывода данных есть функция в Python - … 13
12. Условный оператор в Python - … 26

Проверка практических знаний:

13. Что будет выведено на экран в результате выполнения фрагмента программы
```
a = 16
b = 14
print( ‘a=’,a, ’b=’,b)
```
14. Что будет выведено на экран в результате выполнения фрагмента программы
```
a = 7
b = 9
print ( ‘a=F(‘, b, ‘)’ )
```
15. Что будет выведено на экран в результате выполнения фрагмента программы
```
a = 6
b = 3
print ( ‘F(a)=(b)’ )
```
16. Что будет выведено на экран в результате выполнения фрагмента программы
```
a = 3
b = 13
print( 'F(', a, ')=(', a+b, ')')
```
17. Запишите оператор для вывода значений переменных a=6 и b=7 в формате: 6 + 7 =?
18. Запишите оператор для вывода значений переменных a= 34 и b=56 в формате: F(34)=Q(56)
19. Запишите оператор для вывода значений переменных a=43 и b=33 в формате: a= 43; b= 33
20. Запишите оператор для вывода значений переменных a=23 и b=21 в формате: 23*21=?

Правильные ответы на теоретическую часть теста:

1. Скрипт
2. .py
3. имя/идентификатор, который может принимать некоторое значение.
4. имеет
5. это фрагмент языка программирования, представляющий способ вычисления некоторого значения.
6. комментарий
7. возведение в степень
8. целочисленный, int
9. остаток от деления
10. ввода данных в строку
11. print()
12. if

Приложение № 3

Уровни освоения рабочей программы внеурочной деятельности

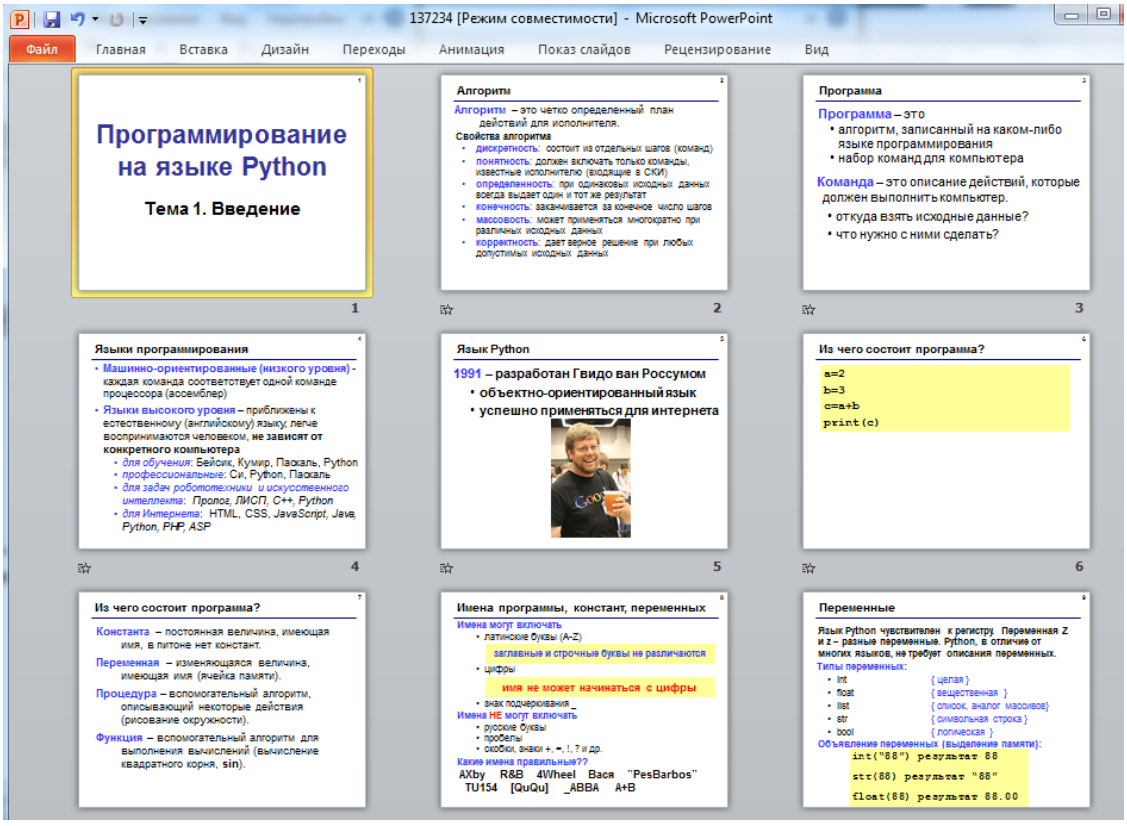
«Основы программирования на Python»

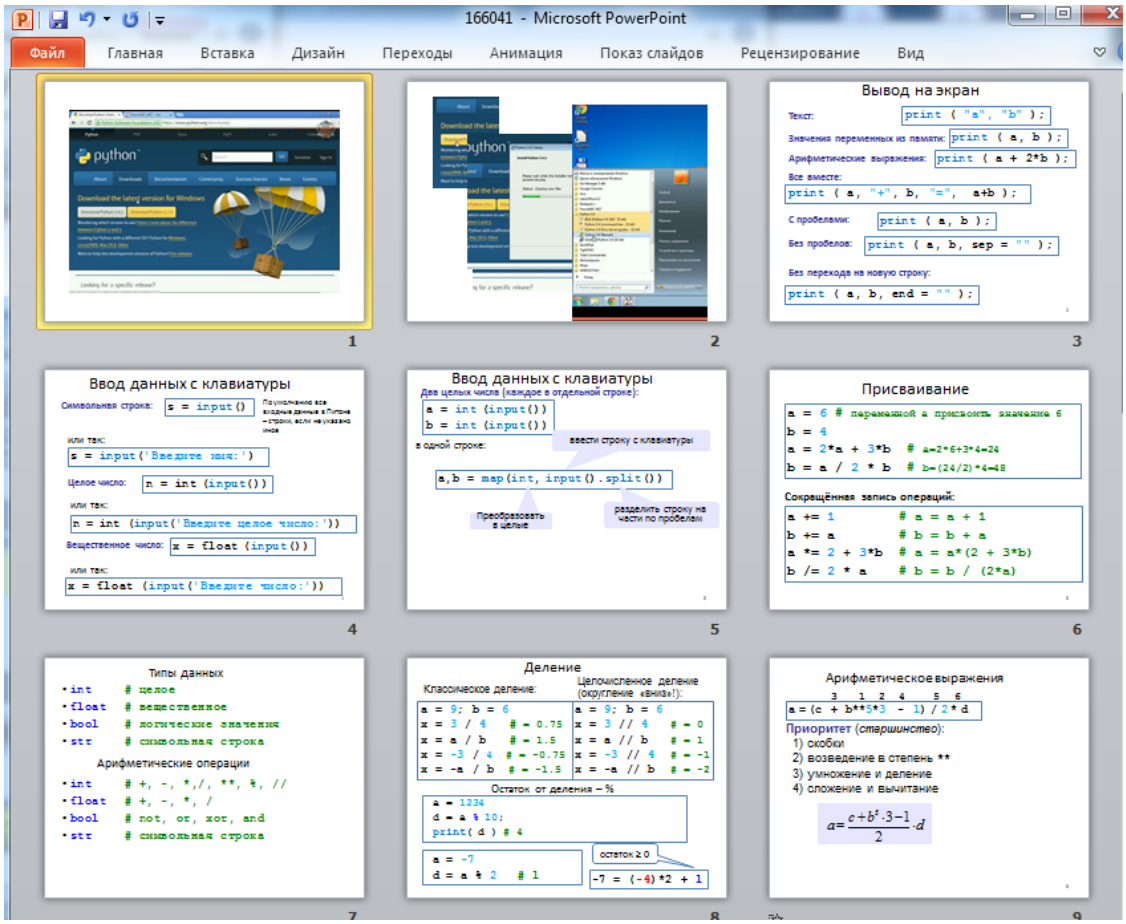
Сводная таблица итоговой аттестации учащихся

№ группы	Количество учащихся	Количество аттестованных учащихся	Уровень освоения ДОП		
			Оптимальный (из них одаренные дети)	Средний	Достаточный
Итого:					

Приложение № 4

Скриншоты презентаций по разным темам курса.





313205 [Режим совместимости] - Microsoft PowerPoint

Файл | Главная | Вставка | Дизайн | Переходы | Анимация | Показ слайдов | Рецензирование | Вид

Программирование на языке Python

Простейшие программы

1

Простейшая программа

```
# Это пустая программа
```

Что делает эта программа?

комментарии после # не обрабатываются

кодировка utf-8 по умолчанию)

```
# coding: utf-8
# Это пустая программа

"""
Это тоже комментарий
"""
```

2

Вывод на экран

```
print ( "2+2=?" )
print ( "Ответ: 4" )
```

автоматический переход на новую строку

Протокол:

```
2+2=?
Ответ: 4
```

```
print ( '2+2=?' )
print ( 'Ответ: 4' )
```

3

Задания

«4а»: Вывести на экран текст «лесенкой»

```
Вася
    пошел
        гулять
```

«5а»: Вывести на экран рисунок из букв

```
   X
  XXX
 XXXXX
XXXXXXX
  XX XX
 XXXXX
```

4

Сложение чисел

Задача. Ввести с клавиатуры два числа и найти их сумму.

Протокол:

```
Введите два целых числа
25
30
25+30=55
```

компьютер

пользователь

компьютер считает сам!

1. Как ввести числа в память?
2. Где хранить введенные числа?
3. Как вычислить?
4. Как вывести результат?

5

Сумма: псевдокод

```
ввести два числа
вычислить их сумму
вывести сумму на экран
```

Псевдокод – алгоритм на русском языке с элементами языка программирования.

Компьютер не может исполнить псевдокод!

6

Переменные

Переменная – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.

7

Имена переменных

МОЖНО использовать

- латинские буквы (A-Z, a-z)
  - заглавные и строчные буквы различаются
- русские буквы (не рекомендуется!)
- цифры
  - имя не может начинаться с цифры
- знак подчеркивания _

НЕЛЬЗЯ использовать

- ~~скобки~~
- ~~знаки +, -, *, / и др.~~

Какие имена правильные?

AXby R&B 4Wheel Вася "PesBarbos"
TU154 [QuQu] _ABBA A+B

8

Типы переменных

```
a = 4
print ( type(a) )
<class 'int'>
```

целое число (integer)

```
a = 4.5
print ( type(a) )
<class 'float'>
```

вещественное число

```
a = "Вася"
print ( type(a) )
<class 'str'>
```

символьная строка

```
a = True
print ( type(a) )
<class 'bool'>
```

логическая

9

Файл | Главная | Вставка | Дизайн | Переходы | Анимация | Показ слайдов | Рецензирование | Вид

Арифметическое выражение

```
      3   1  2   4    5  6
a = (c + b**5*3  - 1) / 2 * d
```

Приоритет (*старшинство*):

1) скобки
2) возведение в степень **
3) умножение и деление
4) сложение и вычитание

$$a = \frac{c + b^5 \cdot 3 - 1}{2} d$$

перенос на следующую строку

```
a = (c + b**5*3  - 1) \
          / 2 * d
```

перенос внутри скобок разрешён

```
a = (c + b**5*3
       - 1) / 2 * d
```

22

Деление

Классическое деление:

```
a = 9; b = 6
x = 3 / 4    # = 0.75
x = a / b    # = 1.5
x = -3 / 4   # = -0.75
x = -a / b   # = -1.5
```

Целочисленное деление (округление «вниз»!):

```
a = 9; b = 6
x = 3 // 4   # = 0
x = a // b   # = 1
x = -3 // 4  # = -1
x = -a // b  # = -2
```

23

Остаток от деления

% – остаток от деления

```
d = 65
b = d // 10
a = d % 10
d = a % b
d = b % a
```

```
a = 15
b = 19
d = a // b
a = a % b
```

24

Операторы // и %

```
a = 1234
d = a % 10; print( d )   4
a = a // 10 ;
d = a % 10; print( d )   3
a = a // 10
d = a % 10; print( d )   2
a = a // 10
d = a % 10; print( d )   1
a = a // 10 ;
```

25

Сокращенная запись операций

```
a += b  # a = a + b
a -= b  # a = a - b
a *= b  # a = a * b
a /= b  # a = a / b
a //= b # a = a // b
a %= b  # a = a % b
```

a += 1

увеличение на 1

26

Ввод двух значений в одной строке

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

21 33 input() — ввести строку с клавиатуры

21 33 input().split() — разделить строку на части по пробелам

целые — применить — эту операцию — к каждой части

21 33 map (int, input().split())

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

27

Задания

«3а»: Ввести три числа: цену пирожка (два числа: рубли, потом – копейки) и количество пирожков. Найти сумму, которую нужно заплатить (рубли и копейки)

Пример:

```
Стоимость пирожка:
12 50
Сколько пирожков:
5
К оплате: 62 руб. 50 коп.
```

«4а»: Ввести число, обозначающее количество секунд. Вывести то же самое время в часах, минутах и секундах.

Пример:

```
Число секунд:
4225
1 ч. 10 мин. 25 с
```

28

Задания

«5а»: Занятия в школе начинаются в 8-30. Урок длится 45 минут, перерывы между уроками – 10 минут. Ввести номер урока и вывести время его окончания.

Пример:

```
Введите номер урока:
5
13-50
```

29

Случайные числа

Случайно...

- встретить друга на улице
- разбить тарелку
- найти 10 рублей
- выиграть в лотерею

Случайный выбор:

- жеребьевка на соревнованиях
- выигравшие номера в лотерее

Как получить случайность?

30

Условный оператор

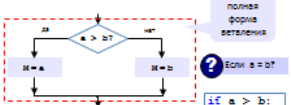
В большинстве реальных задач порядок действий может несколько изменяться, в зависимости от того, какие данные поступили. Например, программа для системы пожарной сигнализации должна выдавать сигнал тревоги, если данные с датчиков показывают повышение температуры или задымленность.
Для этой цели в языках программирования предусмотрены условные операторы, например, для того, чтобы записать в переменную M максимальное из значений переменных a и b, можно использовать оператор:

```
if a > b:
    M = a
else:
    M = b
```

Слово **if** переводится с английского языка как «если», а слово **else** – как «иначе». Если верно (истинно) условие, записанное после ключевого слова **if**, то затем выполняется все команды (*блок команд*), которые расположены *до слова* **else**. *Если же условие после* **if** *неверно* (ложно), выполняются команды, стоящие после **else**.
В Python, в отличие от других языков, важную роль играет 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 (которые вставляются при нажатии на клавишу Tab) или пробелы.

10

Задача: изменить порядок действий в зависимости от выполнения некоторого условия.



полная форма ветвления

Если a = b?

```
if a > b:
    M = a
else:
    M = b
```

отступы

По умолчанию это правило «четырех отступов»

11

Знаки отношений

> <	больше, меньше
>=	больше или равно
<=	меньше или равно
==	равно
!=	не равно

12

Условный оператор

```
if a > b:
    # что делать, если a > b
else:
    # что делать, если a <= b
```

отступы!

```
a = 12
if a > 20:# ложь
  a = 15
print ( a ) # 12
```

```
a = 12
if a > 2:  # истина
  a = 15
else:
  a = 8
print ( a ) # 15
```

13

Внутри условного оператора могут находиться любые операторы, в том числе и другие условные операторы. Например, пусть возраст Андрея записан в переменной **a**, а возраст Бориса – в переменной **b**. Нужно определить, кто из них старше. Одним условным оператором тут не обойтись, потому что есть три возможных результата: старше Андрей, старше Борис и оба одного возраста. Решение задачи можно записать так:

```
if a > b:
    print ( "Андрей старше" )
else:
    if a == b:
        print ( "Одного возраста" )
    else:
        print ( "Борис старше" )
```

Условный оператор, проверяющий равенство, находится внутри блока инструкции (**else**), поэтому он называется *вложенным* условным оператором. Использование вложенных условных операторов позволяет выбрать один из нескольких (*а не только из двух*) вариантов. Если после **else** сразу следует еще один оператор **if**, можно использовать так называемое *каскадное ветвление* с ключевым словом **elif** (сокращение от **else-if**): если очередное условие ложно, выполняется проверка следующего условия и т.д.

```
if a > b:
    print ( "Андрей старше" )
elif a == b:
    print ( "Одного возраста" )
else:
    print ( "Борис старше" )
```

14

Сложные условия

Задача: набор сотрудников в возрасте 25-40 лет (включительно).

сложное условие

```
if v >= 25 and v <= 40 :
  print("подходит")
else:
  print("не подходит")
```

and «И»: одновременное выполнение всех условий!

15

Сложные условия

Задача: набор сотрудников в возрасте 25-40 лет (включительно).

сложное условие

```
if  v < 25 or v > 40
  print("не подходит")
else:
  print("подходит")
```

or «ИЛИ»: выполнение хотя бы одного из двух условий!

16

Цикл – это многократное выполнение одинаковых действий. Доказано, что любой алгоритм может быть записан с помощью трёх алгоритмических конструкций: циклов, условных операторов и последовательного выполнения команд (линейных алгоритмов).

Цикл с условием

```
k = 0
while k < 10:
  print ( "Привет" )
  k += 1
```

При каком условии заканчивает работу?

k >= 10

```
k = 10
while k > 0:
  print ( "Привет" )
  k -= 1
```

При каком условии заканчивает работу?

k <= 0

17

Цикл с переменной

Задача. Вывести 10 раз слово «Привет!».

Можно ли сделать с циклом «пока»?

```
i = 0
while i < 10:
  print("Привет!")
  i += 1
```

Цикл с переменной:

```
for i in range(10):
  print("Привет!")
```

в диапазоне [0,10)

не включая 10!

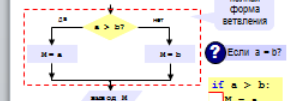
`range(10)` → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

18

37

Условный оператор

Задача: изменить порядок действий в зависимости от выполнения некоторого условия.



полная форма ветвления

Если a = b?

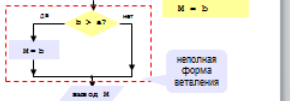
```
if a > b:
    M = a
else:
    M = b
```

отступы

38

Условный оператор: неполная форма

```
M = a
if b > a:
    M = b
```



неполная форма ветвления

Решение в стиле Python:

```
M = max(a, b)
```

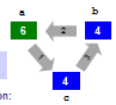
```
M = a if a > b else b
```

39

Условный оператор

```
if a > b:
    c = a
    a = b
    b = c
```

Что делает?



Можно ли обойтись без переменной c?

Решение в стиле Python:

```
a, b = b, a
```

40

Знаки отношений

> <	больше, меньше
>=	больше или равно
<=	меньше или равно
==	равно
!=	не равно

41

Вложенные условные операторы

Задача: в переменных a и b записаны возрасты Андрея и Бориса. Кто из них старше?

Сколько вариантов?

```
if a > b:
  print("Андрей старше")
else:
  if a == b:
    print("Одного возраста")
  else:
    print("Борис старше")
```

Зачем нужен?

вложенный условный оператор

42

Каскадное ветвление

```
if a > b:
  print("Андрей старше")
elif a == b:
  print("Одного возраста")
else:
  print("Борис старше")
```

elif = **else if**

43

Каскадное ветвление

```
cost = 1500
if cost < 1000:
  print ( "Скидок нет." )
elif cost < 2000:
  print ( "Скидка 2%." )
elif cost < 5000:
  print ( "Скидка 5%." )
else:
  print ( "Скидка 10%." )
```

первое сработавшее условие

Что выведет? `Скидка 2%.`

44

Задачи (без функций min и max!)

«3»: Ввести два целых числа, найти наибольшее и наименьшее из них.

Пример:

```
Введите два целых числа:
1 5
Наибольшее число 5
Наименьшее число 1
```

«4»: Ввести четыре целых числа, найти наибольшее из них.

Пример:

```
Введите четыре целых числа:
1 5 4 3
Наибольшее число 5
```

45

Приложение № 5

Задачи для начинающих на языке Python

Задача 1. Проверка является ли число n простым

Пользователь вводит число **n**, программа должна проверить является ли оно простым и сообщить об этом.

Решение:

```
n = int ( input ("Введите число n: "))
for a in range (2, n):
   if (n % a) == 0:
      print ("Число ", n, "не простое")
      break
   elif (n // a) = 1:
      print ("Число ", n, " простое!")
      break
```

Задача 2. Вывод простых чисел в диапазоне d

Вывести список простых чисел в диапазоне d. Диапазон d вводит пользователь.

Решение:

```
d = int ( input ("Введите диапазон: "))
print ("1")
print ("2")
   for n in range (3, d):
      for a in range (2, n):
         if (n % a) == 0:
            break
         else:
            print (n)
            break
```

Задача 3. Вывод таблицы умножения

Вывести таблицу умножения числа M в диапазоне от числа a до числа b. Числа M, a и b вводит пользователь.

Например:

Если M=4, a=2, b=9, то результат будет:

```
4x2=8
4x3=12
4x4=16
4x5=20
4x6=24
4x7=28
4x8=32
4x9=36
```

Решение:

```
M = int( input ("Введите число M: "))    //Запрашиваем у пользователя значения
a = int( input ("Введите диапазон a: "))
b = int( input ("Введите диапазон b: "))
if a <= 0 and b <=0:     //Выполняем проверки
   print ("Ошибка! a и b должны быть > 0!")
elif b<= a:
   print ("Неверный диапазон! b должно быть больше a!")
else:
```

```
print ("Результат:")
for a in range (a-1, b):     //Выводим результат циклом
    a += 1
    print (M, "x", a, "=", M * a)
```

Задача 4. Високосный год

Часть 1. Написать функцию, которая определяет, является ли год високосным. Пользователь вводит год, если он високосный, то функция возвращает True. Если нет, то False.

Решение:

```
def year_leap (y):
   if y % 4 == 0:
      return True
   else:
      return False
s = year_leap (int (input "Введите год: "))
print (s)
```

Часть 2. Вывести список високосных годов в заданном диапазоне. Пользователь вводит диапазон годов, программа выводит список високосных годов в данном диапазоне.

Решение:

```
frm = int (input ("Введите год от: "))
to = int (input ("Введите год до: "))
if frm >= to or frm == 0:
   print ("Неверный диапазон!")
   exit ()
for year in range (frm, to + 1):
   if year % 4 == 0:
      print (year)
      year += 1
```

Задача 5. Площадь, периметр и диагональ квадрата

Написать функцию, которая выводит периметр, площадь и диагональ квадрата, после того как пользователь вводит сторону.

Решение:

```
def square (sd):
   return (4*sd, sd**2, (2*sd**2)**.5)
result = square (int (input ("Введите сторону квадрата: ")))
print (result)
```

Задача 6. Простейший калькулятор на Python

Написать простейший калькулятор. Пользователь вводит число **a**, число **b** и вид арифметической операции над ними. Функция возвращает результат.

Решение:

```
def arithmetic (a, b, c):
   if c == "+":
      return a + b
   elif c == "-":
      return a - b
   elif c == "*":
      return a * b
   elif c == "/":
      return a / b
```

```
    else:
        return "Неизвестная операция!"
result = arithmetic (int(input("Введите число a")), int(input("Введите число b")), input(
"Введите операцию:"))
print("Результат: ", result)
```

Задача 7. Времена года

Пользователь вводит номер месяца, функция возвращает время года.

Решение:

```
def season (m):
        if m == 12 or m == 1 or m == 2:
                return ”Это зима”
        elif m == 3 or m == 4 or m == 5:
                return ”Это весна”
        elif m == 6 or m == 7 or m == 8:
                return ”Это лето”
elif m == 9 or m == 10 or m == 11:
                return ”Это осень”
elif m == 0 or m > 12:
                return ”Нет такого месяца!”
result = season (int(input(“Введите номер месяца: ”)))
print (result)
```

Задача 8. Простой калькулятор банковского вклада

Пользователь вносит указанную сумму на указанное количество лет под 10% годовых. Вернуть сумму, которая будет на счете в конце срока. Необходимо учесть, что проценты начисляются на остаток на счете каждый год.

Решение:

```
def bank (a, years):
    for i in range (years):
        a = a + (a/100*10)
    return (a)
result = bank (float(input("Введите сумму вклада: ")), int(input("На сколько лет: ")))
print (result)
```

Задача 9. Самое длинное слово в предложении

Пользователь вводит предложение(какой-либо текст). Программа вычисляет самое длинное слово в предложении.

Решение:

```
s = input ("Введите ваш текст: ")
w = max(s.split(), key=len)
print ("Самое длинное слово в предложении: ", w)
```

Подробный разбор решения задачи «Високосный год или нет» для объяснения учащимся на занятии

Вводится год. Определить, является ли он високосным или обычным.

Примечание. Високосными являются года, которые делятся на 4, за исключением столетий, которые не делятся на 400.

Листинг программы (скрипт):

```
year = int(input())
if year % 4 != 0:
    print("usual year")
```

```
elif year % 100 == 0:
    if year % 400 == 0:
        print("intercalary year")
    else:
        print("usual year")
else:
    print("intercalary year")
```

Короткое решение:

```
year = int(input())
if year % 4 != 0 or (year % 100 == 0 and year % 400 != 0):
    print("usual year")
else:
    print("intercalary year")
```

Решение с комментариями:

```
# Вводится год, преобразуется к целому числу
year = int(input())

# Если остаток от деления на 4 не равен нулю,
# значит год не делится нацело на 4 и
# не является високосным, т. е. он обычный.
if year % 4 != 0:
    print("usual year")
# Исключаем столетия, которые не делятся на 400
elif year % 100 == 0: # является ли столетием?
    if year % 400 == 0: # Делится ли на 400?
        # В таком случае год високосный
        print("intercalary year")
    else: # Если столетие, но не делится на 400, то год обычный
        print("usual year")
# Во всех остальных случаях год високосный
else:
    print("intercalary year")
```

Короткое решение с комментариями:

```
year = int(input())
# Сразу проверяются все условия.
# Если год не делится на 4 или делится на 100, но не на 400,
# то он обычный. Во всех остальных случаях - високосный.
if year % 4 != 0 or (year % 100 == 0 and year % 400 != 0):
    print("usual year")
else:
    print("intercalary year")
```

Пример выполнения скрипта:

```
2024
intercalary year
```

Шпаргалка по Python 3

(Официальная документация по Python 3 http://docs.python.org/py3k/ Перевод В. Федотов)

Базовые типы

integer, float, boolean, string

```
  int  783    0    -192
float  9.23   0.0   -1.7e-6        (e-6 = 10^-6)
 bool  True   False
  str  "One\nTwo"   'I\'m'
```

"One\nTwo": перевод строки; 'I\'m': экранирована

неизменяемая, упорядоченная последовательность символов

многострочные:
```
"""X\tY\tZ
1\t2\t3"""
```
символ табуляции

Контейнерные типы

▪ упорядоченная последовательность, быстрый доступ по индексу

```
  list  [1,5,9]   ["x",11,8.9]   ["word"]   []
 tuple  (1,5,9)   11,"y",7.4     ("word",)  ()
```

неизменяемые

выражение с одними запятыми

str — как упорядоченная последовательность символов

▪ порядок заранее неизвестен, быстрый доступ по ключу, ключи = базовые типы или кортежи

```
  dict  {"key":"value"}                                {}
```

словарь

```
        {1:"one",3:"three",2:"two",3.14:"π"}
```

соответствие между ключами и значениями

```
   set  {"key1","key2"}   {1,9,3,0}   set()
```

Имена

для переменных, функций, модулей, классов...

a..zA..Z_ потом a..zA..Z_0..9

- нелатинские буквы разрешены, но избегайте их
- ключевые слова языка запрещены
- маленькие/БОЛЬШИЕ буквы отличаются

☺ `a toto x7 y_max BigOne`

☹ ~~`8y and`~~

Преобразования

`type(`*выражение*`)`

`int("15")` можно указать целое основание системы исчисления вторым параметром

`int(15.56)` отбросить дробную часть (для округления делайте `round(15.56)`)

`float("-11.24e8")`

`str(78.3)` и для буквального преобразования → `repr("Text")`

см. форматирование строк на другой стороне для более тонкого контроля

`bool` → используйте сравнения (==, !=, <, >, ...), дающие логический результат

`list("abc")` → *использует каждый элемент последовательности* → `['a','b','c']`

`dict([(3,"three"),(1,"one")])` → `{1:'one',3:'three'}`

`set(["one","two"])` → *использует каждый элемент последовательности* → `{'one','two'}`

`":".join(['toto','12','pswd'])` → `'toto:12:pswd'`

соединяющая строка; последовательность строк

`"words with spaces".split()` → `['words','with','spaces']`

`"1,4,8,2".split(",")` → `['1','4','8','2']`

строка-разделитель

Присвоение переменным

```
x = 1.2+8+sin(0)
```

значение или вычислимое выражение

имя переменной (идентификатор)

```
y,z,r = 9.2,-7.6,"bad"
```

имена переменных; контейнер с несколькими значениями (здесь кортеж)

`x+=3` ← добавление; вычитание → `x-=2`

`x=None` «неопределённая» константа

Доступ к элементам последовательностей

для списков, кортежей, строк...

отрицательный индекс	-6	-5	-4	-3	-2	-1
положительный индекс	0	1	2	3	4	5
	lst=[11,	67,	"abc",	3.14,	42,	1968]
положительный срез	0	1	2	3	4	5 6
отрицательный срез	-6	-5	-4	-3	-2	-1

`len(lst)` → `6`

доступ к отдельным элементам через [индекс]

`lst[1]`→`67` `lst[0]`→`11` *первый*

`lst[-2]`→`42` `lst[-1]`→`1968` *последний*

доступ к подпоследовательности [*начало среза*:*конец среза*:*шаг*]

```
lst[:-1]→[11,67,"abc",3.14,42]
lst[1:-1]→[67,"abc",3.14,42]
lst[::2]→[11,"abc",42]
lst[:]→[11,67,"abc",3.14,42,1968]
```

срез без указания границ → с начала до конца

```
lst[1:3]→[67,"abc"]
lst[-3:-1]→[3.14,42]
lst[:3]→[11,67,"abc"]
lst[4:]→[42,1968]
```

Для изменяемых последовательностей, полезно удаление `del lst[3:5]` *и изменение с помощью присвоения* `lst[1:4]=['hop',9]`

Булева логика

Сравнения: `< > <= >= == !=` (≤ ≥ = ≠)

`a and b` логическое и — *оба верны одновременно*

`a or b` логическое или — *верно хотя бы одно*

`not a` логическое нет

`True` константа «истина»

`False` константа «ложь»

Блоки инструкций

родительская инструкция :

блок инструкций 1...

родительская инструкция :

блок инструкций 2...

отступы !

след. инструкция после блока 1

Условный оператор

выражения в блоке выполняется только если условие истинно

`if` *логическое выражение* :

→ *блок выражений*

может сопровождаться несколькими elif, elif, ..., но только одним окончательным else. Пример:

```
if x==42:
    # блок выполнится, если x==42 истинно
    print("real truth")
elif x>0:
    # иначе блок, если лог. выражение x > 0 истинно
    print("be positive")
elif bFinished:
    # иначе блок, если лог. перем. bFinished истинна
    print("how, finished")
else:
    # иначе блок для всех остальных случаев
    print("when it's not")
```

Математика

числа с плавающей точкой... приближенные значения!

углы в радианах

Операторы: `+ - * / // % **` (× ÷ ↑ ↑ a^b)

деление без остатка; остаток

```
(1+5.3)*2→12.6
abs(-3.2)→3.2
round(3.57,1)→3.6
from math import sin,pi...
sin(pi/4)→0.707...
cos(2*pi/3)→-0.4999...
acos(0.5)→1.0471...
sqrt(81)→9.0   √
log(e**2)→2.0  и т.д. (см. доки)
```

Цикл с условием

блок инструкций выполняется до тех пор, пока условие истинно

```
while логическое выражение :
    блок инструкций
```

```
s = 0
i = 1                    # инициализации перед циклом
while i <= 100:          # условие с хотя бы одним изменяющимся значением (здесь i)
    # выражения вычисляются пока i ≤ 100
    s = s + i**2
    i = i + 1            # изменяет переменную цикла
print("sum:",s)          # вычисленный результат цикла
```

$s = \sum_{i=1}^{i=100} i^2$

⚠ остерегайтесь бесконечных циклов !

Управление циклом

```
break       немедленный выход
continue    следующая итерация
```

Цикл перебора

блок инструкций выполняется для всех элементов контейнера или итератора

```
for переменная in подпоследовательность :
    блок инструкций
```

Проход по **элементам** последовательности

```
s = "Some text"          # инициализации перед циклом
cnt = 0
for c in s:              # переменная цикла, значение управляется циклом for
    if c == "e":         # Посчитать число букв e в строке
        cnt = cnt + 1
print("found",cnt,"'e'")
```

цикл по dict/set = цикл по последовательности ключей

используйте срезы для проходов по подпоследовательностям

Проход по **индексам** последовательности

- можно присваивать элемент по индексу
- доступ к соседним элементам

```
lst = [11,18,9,12,23,4,17]
lost = []
for idx in range(len(lst)):
    val = lst[idx]
    if val > 15:         # Ограничить значения больше 15, запомнить потеряные значения
        lost.append(val)
        lst[idx] = 15
print("modif:",lst,"-lost:",lost)
```

Пройти одновременно по **индексам** и **значениям** :

```
for idx,val in enumerate(lst):
```

Печать / Ввод

```
print("v=",3,"cm :",x,",",y+4)
```

элементы для отображения : литералы, переменные, выражения

настройки `print`:

- `sep=" "` (разделитель аргументов, по умолч. пробел)
- `end="\n"` (конец печати, по умолч. перевод строки)
- `file=f` (печать в файл, по умолч. стандартный вывод)

```
s = input("Instructions:")
```

⚠ `input` всегда возвращает **строку**, преобразуйте её к нужному типу сами (см. «Преобразования» на другой стороне).

Операции с контейнерами

```
len(c)  → количество элементов
min(c)  max(c)  sum(c)
sorted(c) → отсортированная копия
val in c → boolean, membership operator in (absence not in)
enumerate(c) → итератор по парам (индекс, значение)
```

*Прим. : для словарей и множеств эти операции работают с **ключами**.*

*Только для **последовательностей** (lists, tuples, strings) :*

```
reversed(c) → reverse iterator     c*5 → повторить     c+c2 → соединить
c.index(val) → позиция             c.count(val) → подсчёт вхождений
```

Генераторы последовательностей int

часто используются в циклах for

`range([start,]stop[,step])` — по умолчанию 0; не включается

```
range(5)      → 0 1 2 3 4
range(3,8)    → 3 4 5 6 7
range(2,12,3) → 2 5 8 11
```

`range` возвращает «генератор», чтобы увидеть значения, преобразуйте его в последовательность, например:
`print(list(range(4)))`

Операции со списками

⚠ изменяют первоначальный список

```
lst.append(item)     добавить элемент в конец
lst.extend(seq)      добавить последовательность в конец
lst.insert(idx,val)  вставить значение по индексу
lst.remove(val)      удалить первое вхождение val
lst.pop(idx)         удалить значение по индексу и вернуть его
lst.sort()  lst.reverse()   сортировать/обратить список по месту
```

Определение функций

имя функций (идентификатор); именованные параметры

```
def fctname(p_x,p_y,p_z):
    """documentation"""
    # инструкции, вычисление результата
    return res           # результат вызова. если нет возврата значения, по умолчанию вернёт None
```

⚠ параметры и всесь этот блок существуют только *во время* вызова функции («черная коробка»)

Вызов функций

```
r = fctname(3,i+2,2*i)   # один аргумент каждому параметру
```

получить результат (если нужен)

Операции со словарями

```
d[key]=value     d.clear()
d[key]→value     del d[key]
d.update(d2)     # Обновить/добавить пары
d.keys()         # просмотр ключей,
d.values()       # значений и пар
d.items()        # просмотр ключей,
d.pop(key)       # значений и пар
```

Операции с множествами

Операторы:

```
|  → объединение (вертикальная черта)
&  → пересечение
-  ^  → разность/симметричная разн.
<  <=  >  >=  → отношения включения
s.update(s2)
s.add(key)  s.remove(key)
s.discard(key)
```

Файлы

Сохранение и считывание файлов с диска

```
f = open("fil.txt","w",encoding="utf8")
```

- файловая **переменная** для операций
- **имя** файла на диске (+путь…)
- **режим** работы: `'r'` read, `'w'` write, `'a'` append…
- **кодировка** символов в текстовых файлах: `utf8` `ascii` `cp1251` …

см. функции в модулях `os` и `os.path`

запись

```
f.write("hello")
```

⚠ текстовый файл → *чтение/запись только **строк**, преобразуйте требуемые типы*

чтение

```
s = f.read(4)        # пустая строка при конце файла; если количество символов не указано, прочитает весь файл
s = f.readline()     # прочитать следующую строку
```

```
f.close()
```

⚠ не забывайте закрывать после использования

Автоматическое закрытие : `with open(…) as f:`

очень часто: цикл по строкам (каждая до '\n') текстового файла

```
for line in f :
    # блок кода для обработки строки
```

Форматирование строк

форматные директивы; значения для форматирования

```
"model {} {} {}".format(x,y,r)  → str
"{селектор:формат!преобразование}"
```

- **Селекторы** :
  - `2`
  - `x`
  - `0.nom`
  - `4[key]`
  - `0[2]`
- **Формат** :

заполнение выравнивание знак минимирина . точность~максимирина тип

`< > ^ =` `+ - пробел` `0` в начале для заполнения 0

целые: `b` бинарный, `c` символ, `d` десятичн. (по умолч.), `o` 8-ричн, `x` или `X` 16-ричн.

float: `e` or `E` экспоненциальная запись, `f` or `e` фиксир. точка,
`g` or `G` наиболее подходящая из `e` или `F`, `%` перевод долей в %

строки : `s` …

- **Преобразование** : `s` (читаемый текст) или `r` (в виде литерала)

Примеры:

```
"{:+2.3f}".format(45.7273)
→'+45.727'
"{1:>10s}".format(8,"toto")
→'      toto'
"{!r}".format("I'm")
→'"I\'m"'
```

Приложение № 6

Упражнения, уменьшающие усталость при работе за компьютером

Упражнения для улучшения осанки

«Глядя в небо»

Цель упражнения: устранение вредных эффектов от неподвижного сидения в течение длительного периода времени и профилактика грыжи межпозвоночных дисков поясничного отдела.

Упражнение: стоя, руки лежат на бедрах. Медленно отклоняться назад, глядя в небо. Вернуться в исходное положение.

«Египтянин»

Цель упражнения: укрепление мышц задней стороны шеи для улучшения осанки и предотвращения болей в области шеи.

Упражнение способствует предотвращению:

- синдрома запястного канала;
- вытягиванию шеи вперед;
- дисфункции височно-нижнечелюстного сустава;
- грыжи межпозвоночных дисков шейного отдела;
- синдрома верхней апертуры грудной клетки.

Упражнение: сидя или стоя, взгляд направлен прямо, а не вверх и не вниз.

Надавив указательным пальцем на подбородок, сделать движение шеей назад.

В этом положении следует оставаться в течение 5 секунд.

«Абра-кадабра»

Цели упражнения:

- усиление кровотока к ладоням;
- снятие напряжения в запястьях и ладонях;
- удаление продуктов распада из области запястного канала и ладоней.

Упражнение: сидя, руки лежат на подлокотниках, запястья должны быть вытянуты ладонями вниз.

Абра: медленно сжать ладони в кулак.

Кадабра: медленно разжать кулаки.

Для достижения желаемого результата эти упражнения следует повторять не менее 10 раз.

Упражнения для глаз

Эффективная профилактическая мера - зрительная гимнастика. Ее проводят дважды: через 7-8 минут от начала работы ребенка на компьютере и после ее окончания. Непродолжительная гимнастика - около одной минуты - проста и доступна каждому. Например, сидя за компьютером, ребенок поднимает глаза кверху и, представив летящего там мотылька или бабочку, следит за их полетом из одного угла комнаты в другой, не поворачивая при этом головы - двигаться должны только глаза!

Есть, конечно, и другие нехитрые правила:

1. На счет 1-4 закрыть глаза, не напрягая глазные мышцы, на счет 1-6 широко раскрыть глаза и посмотреть вдаль. Повторить 4-5 раз.

2. Посмотреть на кончик носа на счет 1-4, а потом перевести взгляд вдаль на счет 1-6. Повторить 4-5 раз.

3. Не поворачивая головы, медленно делать круговые движения глазами вверх-вправо-вниз-влево и в обратную сторону: вверх-влево-вниз-вправо. Затем посмотреть вдаль на счет 1-6. Повторить 4-5 раз.

4. Держа голову неподвижно, перевести взор, зафиксировав его, на счет 1-4 вверх, на счет 1-6 прямо; затем аналогично вниз-прямо, вправо-прямо, влево-прямо. Проделать движение по диагонали в одну и другую стороны, переводя глаза прямо на счет 1-6. Повторить 3-4 раза.

5. Не поворачивая головы, закрытыми глазами «посмотреть» направо на счет 1-4 и прямо на счет 1-6. Поднять глаза вверх на счет 1-4, опустить вниз на счет 1-4 и перевести взгляд прямо на счет 1-6. Повторить 4-5 раз.

6. Посмотреть на указательный палец, удаленный от глаз на расстоянии 25-30 см, и на счет 1-4 приблизить его к кончику носа, потом перевести взор вдаль на счет 1-6. Повторить 4-5 раз.